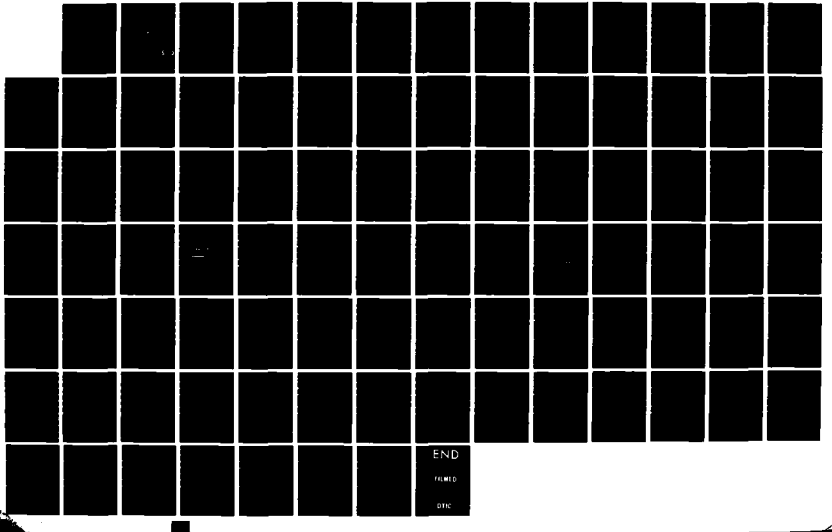
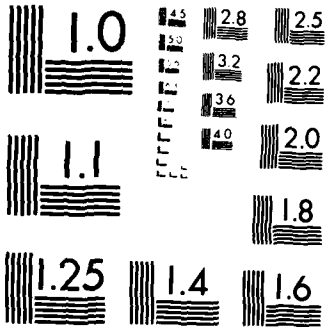
AD-A151 761 GRAPHIC SIMULATION OF A MACHINE-REPAIRMAN MODEL(U) 1/1
NAVAL POSTGRADUATE SCHOOL MONTEREY CA R E NELSEN
SEP 84

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

GRAPHIC SIMULATION OF A MACHINE-REPAIRMAN MODEL

by

Rex Ernest Nelsen

September 1984

Thesis Advisor: James D. Esary

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. A157 761
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Graphic Simulation Of a Machine-Repairman Model	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1984
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Rex Ernest Nelsen	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943	12. REPORT DATE September 1984
	13. NUMBER OF PAGES 87
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DTIC COPY INSPECTED 1

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Graphic simulation
Microcomputer simulation
Birth-death process
Machine-repairman model

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A discrete-event simulation of a stochastic process, a machine-repairman model, has been programmed on the IBM Personal Computer. The model consists of three helicopters, of which two are in service and one is in cold standby, with an option of one or two repairmen.

The program output is a graphics display containing a system state-versus-time graph, a table of statistics, and animated figures that illustrate the current state of the system. The program user can directly

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (Cont.)

observe the dynamics of the model as the fixed-increment, simulation clock advances. The user has the option of changing the following model parameters: helicopter failure rate, repairman service rate, and the number of repairmen to employ.

Approved for public release, distribution unlimited

Graphic Simulation of a Machine-Repairman Model

by

Rex Ernest Nelsen
Major, United States Marine Corps
B.S., University of Idaho, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September, 1984

Author: Rex E Nelsen
Rex E. Nelsen

Approved by: V. Esary
James D. Esary, Thesis Advisor

Alvin F. Andrus, Second Reader

E T Howard acting
Alan R. Washburn, Chairman,
Department of Operations Research

Kneale T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

A discrete-event simulation of a stochastic process, a machine-repairman model, has been programmed on the IBM Personal Computer. The model consists of three helicopters, of which two are in service and one is in cold standby, with an option of one or two repairmen.

The program output is a graphics display containing a system state-versus-time graph, a table of statistics, and animated figures that illustrate the current state of the system. The program user can directly observe the dynamics of the model as the fixed-increment, simulation clock advances. The user has the option of changing the following model parameters: helicopter failure rate, repairman service rate, and the number of repairmen to employ.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENT

The author wishes to thank Professors James D. Esary and Alvin F. Andrus for their suggestions and encouragement during this project. Special thanks must go to Professor Bruno O. Shubert for the generous contribution of his assembly-language random-number generator. I am indebted to Professor Gordon E. Latta and Lieutenant Commander Jeffrey E. Ferris, U.S. Navy, for assisting me with the totally unfamiliar IBM Personal Computer. I especially wish to thank my wife Suzanne whose encouragement, support, and understanding contributed enormously to a successful, two-year graduate program.

I. INTRODUCTION

During recent years, the versatile microcomputer has found a niche in many of the professional discliplines including education. For difficult and complex subjects, educators are exploring the possibilities of the microcomputer as a teaching aid in addition to its more traditional role as a computing machine. Some universities now use microcomputer graphics to illustrate principles that cannot be fully explained in a lecture. [Ref. 1]

A widely used application of computers is to imitate or simulate stochastic processes. This technique usually involves only the numerical evaluation over time of the corresponding model with the intent of gaining some understanding of the process. But using graphics as part of the output in a simulation provides the user another perspective. For example, observing a working model--such as a birth-death process--can help in grasping the fundamentals of the model and provide some insights into the process not easily seen on the classroom chalkboard.

This paper describes the development of a discrete-event, graphic simulation of a specific example of the machine-repairman model. The simulation gives the observer the impression of real-time passage. A graph on the visual display shows the history of the system's transitions from

state to state, and animated figures illustrate the dynamics of the model. The simulation also exhibits a table of the theoretical, and the current, estimated values of quantities commonly used in evaluating queueing systems.

Section II of this paper is a discussion of the machine-repairman model, the assumptions upon which the model is based, and the methods used for obtaining the theoretical and estimated values of the quantities of interest. Section III covers the simulation's implementation on the IBM Personal Computer (IBM PC). Appendix A is a user's guide which contains summary information about the machine-repairman model and instructions for operating the program. The program flowchart and the program listings are found in Appendixes B, C, D, and E.

II. THE MACHINE-REPAIRMAN MODELS

A. DISCUSSION

The simulation described in this paper actually contains two machine-repairman models. In both models there are three machines, represented by helicopters. The models are distinguishable by the number of repairmen; there is one repairman in the first and two repairmen in the second.

The nominal mission is to keep two helicopters flying at all times. The third helicopter is placed in cold standby as a backup to the other two. If available, a repairman is assigned immediately to a helicopter upon its failure. If a repairman is not free, then the helicopter joins a queue to await repair.

Several assumptions are made to simplify the models. In the two-repairmen model, repairmen are assumed to be equally competent and to take the same mean time to repair a helicopter. Both models assume the repair time to be continuous; that is, repair parts are always available, so there is no delay in repairing a machine other than the time required by the repairman to perform the work. The repairmen work independently and do not assist one another. Similarly, the models assume the helicopters operate independently and have the same service life with the same mean time to failure. By assumption, a helicopter is not

language subroutines, generating the graphics display, and performing the tasks selected from the program menu. Within the main program are routines for computing the limiting values of the tabulated quantities and for printing them on the display screen. The main program also calculates the initial failure times for the first entry into the clock module.

The clock module determines the next-event time, whether the event is a failure or the completion of repair, and which helicopter it affects. The simulation clock is a fixed-increment, time-advance type. In contrast to a next-event, time-advance clock which skips over the intervening time between events, the clock module in MACHREPR advances the time in equal increments. With this type of clock, the user experiences the illusion of real-time passage.

During the execution of the simulation, the clock module monitors the keyboard for certain commands issued by the user. Pressing the "P" key pauses the program to allow the user to study the graphic display. Pressing the "C" key causes the program to continue. The "S" key will stop the simulation and return the program to the program menu. Should the clock reach the preset limit of 9950 time units, the program will automatically stop the simulation and return to the program menu.

Every time the clock is advanced one increment, the clock module updates the system state-versus-time graph

seen above the others is in active service; the two machines with broken rotors are down. In this example, the helicopter graphics indicate that the system is in state 2, where helicopter B is flying, helicopter C is under repair, and helicopter A is awaiting repair. The graph shows that the current time is 69 and that the system jumped from state 1 to 2 at t = 68.

The table in the lower right corner contains the estimated and the limiting values of the quantities discussed in Section II. The row labels are:

Ar - the arrival rate of machines into the repair queue,

Wq - the average wait time of a machine in the repair queue,

Lq - the average number of machines in the repair queue,

W - the average amount of time a machine is down,

L - the average number of machines in the repair system,

Avl - the proportion of time that at least one machine is available.

Pd - the probability that a down machine must wait for repair,

Ps - the probability that an up machine must go to cold standby.

MACHREPR consists of four major parts: (1) main program, (2) clock module, (3) failure module, and (4) repair module. The main program handles the housekeeping details such as initializing variables, loading the machine

generator seed, or (5) end the program. The instructions that are available on the screen are a summarized version of those found in Appendix A, the user's guide. The model parameters that can be changed by the user are the mean time to failure, the mean time for repair, and the number of repairmen.

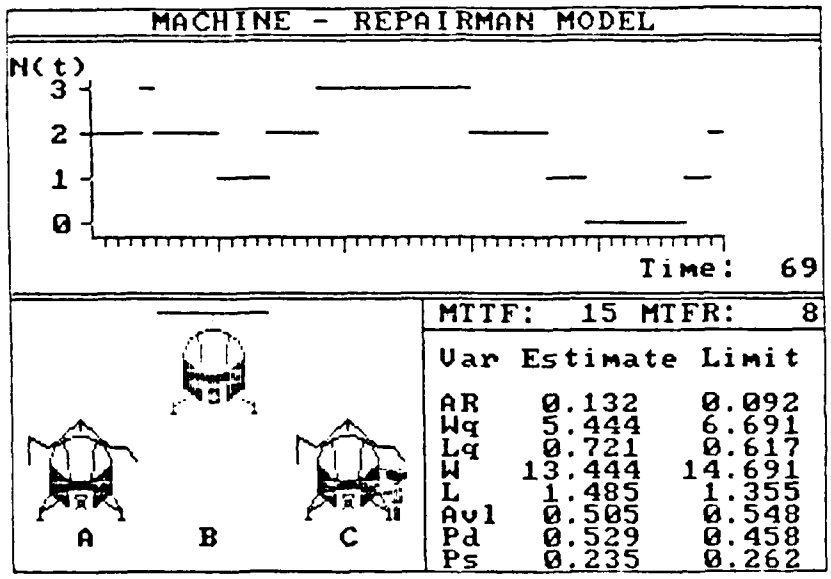


Figure 3: An Example of the Graphics Display.

Figure 3 shows an example of the graphics display screen as it appears during the simulation. The upper half of the screen contains the system state-versus-time graph. The system state values are on the ordinate of the graph, and fifty divisions representing time are on the abscissa. The three graphic figures of helicopters in the lower left corner symbolize the status of the system. The helicopter

B. THE PROGRAMMING LANGUAGES

MACHREPR is written in Advanced BASIC which is the programming language supplied with PC-DOS. Advanced BASIC was selected because of its convenient graphics commands and its ready availability. Although Advanced BASIC is an interpreter, program speed was not an essential criterion that warranted using a compiled language. Advanced BASIC was insufficient, however, in two aspects, and so MACHREPR uses two subroutines written in 8088 assembly language to improve program performance. These two subroutines, named SCRNSHFT and RNGEN, will be described later in this section.

C. PROGRAM DESCRIPTION

Appendix B contains the flowcharts for the BASIC program MACHREPR. The flowcharts illustrate the simulation's general construction and program flow; however, they are not a verbatim duplication of the program listing. The program listing for MACHREPR and the two source-code listings for RNGEN and SCRNSHFT are in Appendixes C, D, and E respectively.

After initializing several variables and arrays, the program presents the title screen. The program menu appears next with the following options: (1) see the program instructions, (2) change the model's parameters, (3) use the default parameters, (4) set the random-number

III. THE SIMULATION

A. THE COMPUTER

The microcomputer used for this simulation was the IBM Personal Computer. Since the program MACHREPR uses color graphics extensively, the computer requires a color display monitor and either IBM's Color/Graphics Adapter or a suitable graphics card from another manufacturer. The program will not run without a graphics adapter. Other minimum, hardware requirements for this simulation are 128 kilobytes of read-write memory (RAM) and at least one 5¼-inch, floppy disk-drive.

At the beginning of this project, the author chose to use IBM's disk operating system PC-DOS with the intent of enabling the simulation to run on other microcomputers using the more generic version of PC-DOS: MS-DOS by Microsoft. It soon became apparent, however, that MACHREPR would need to directly access certain parts of the computer's memory in order to improve the program's performance. Since the program makes direct calls to the PC-DOS, there can be no assurance that the simulation will run on other "IBM-compatible" machines.

For $\hat{P}_s$, the only time a machine must wait in cold standby is when the system is in state 0; therefore, $\hat{P}_s = \hat{p}_3$.

The availibility of at least one helicopter in the system is the sum over all states n of the percentage of the flyable helicopters in each state multiplied by the proportion of time the system is in that state;

$$Av = p_0(1) + p_1(2/3) + p_2(1/3) + p_3(0).$$

The estimate of the availabiliy is calculated by substituting $\hat{p}_n$, n = 0,1,2,3, into the above expression.

The average waiting time quantities, W and W_q are then calculated by using Little's formulas

$$W = L/\lambda$$

$$W_q = L_q/\lambda.$$

To compute the estimate $\hat{W}_q$, the program keeps a record of the length of time each helicopter spends in the repair queue and uses the expression

$$\hat{W}_q = (1/m) \sum_{i=1}^{m} q_i \qquad m = 0,1,2,...$$

where q_i is the waiting time in queue of the ith helicopter and m is the total number of helicopters that have entered the queue by time t. Then $\hat{W}$ is easily found with

$$\hat{W} = \hat{W}_q + E(S).$$

The estimate $\hat{P}_d$ of the probability that a down machine must wait for repair is the proportion of the time that the system spends in a state in which a machine is down with no unemployed repairman. The repairman is not available in the one-repairman model when the system is in state 2 or 3:

$$\hat{P}_d = \hat{p}_2 + \hat{p}_3.$$

In the two-repairmen case, the repairmen are busy when the system is in state 3:

$$\hat{P}_d = \hat{p}_3.$$

In the two-repairmen case, a queue forms only when the system is in state 3, thus

$$L_q = p_3, \text{ and}$$

$$\hat{L}_q = \hat{p}_3.$$

The average rate at which helicopters enter the repair queue, λ, is dependent upon the number of active machines subject to failure in state n and the proportion of time the system is in state n. There are two active helicopters in states 0 and 1, one in state 2, and none in state 3. The combined failure rate of the active helicopters in state n is given by

$$\alpha_n = \begin{cases} 2\alpha & n = 0,1 \\ \alpha & n = 2 \\ 0 & n = 3. \end{cases}$$

Then λ is found by

$$\lambda = \sum_{n=0}^{3} \alpha_n p_n = 2\alpha p_0 + 2\alpha p_1 + \alpha p_2.$$

For the estimate $\hat{\lambda}$, the program divides the number of failures that have occurred in time t by t, that is

$\hat{\lambda}$ = total number of failures/total elapsed time.

failure rate and the repairman service rate, will not be discussed in this paper. The interested reader should refer to the bibliography for a list of textbooks on queueing theory and birth-death processes. In addition to the limiting probabilities, $\{p_n, n = 0,1,2,3\}$, the program computes the fraction of the time the system is in state n, denoted $\hat{p}_n$, as equal to the total time in state n divided by the total elapsed time. The quantity $\hat{p}_n$ is an estimate of p_n.

The value of L, the theoretical average number of machines in the repair system, is found by averaging the number of down machines over all states:

$$L = \sum_{n=0}^{3} np_n = (1)p_1 + (2)p_2 + (3)p_3.$$

The estimate of L, denoted $\hat{L}$, is computed with the same expression with the exception that the estimate $\hat{p}_n$ is substituted for p_n.

The average number of machines in the repair queue L_q and its estimate $\hat{L}_q$ depends in part upon the number of repairmen in the model. In the one-repairman case, machines occupy the repair queue only when the system is in state 2 or 3, thus

$$L_q = p_2 + 2p_3, \text{ and}$$

$$\hat{L}_q = \hat{p}_2 + 2\hat{p}_3.$$

As shown in Figures 1 and 2, the variable q is the time that a down machine waits in the repair queue. Then $W_q = E(q)$. It follows that the average total time that a helicopter is in the repair system is the sum of the average waiting time in queue and the expected service time: $W = W_q + E(S)$.

Little's formulas show the relationship of the four quantities of interest

$$L = \lambda W$$

$$L_q = \lambda W_q$$

where λ is defined as the average arrival rate of machines entering the repair-queue system.

There are three additional quantities that will be computed and presented by the simulation. They are defined as

P_d = P{down machine must wait for repair},

P_s = P{up machine must go to cold standby},

Av = availability = the proportion of time that at least one helicopter is not down.

To calculate all of eight of these values, it is first necessary to compute the system's steady-state probabilities, $p_n = P\{N = n\}$, $n = 0,1,2,3$. In other words, p_n is the long-run probability that the system is in state n. The details of calculating p_n, a function of the machine

of a helicopter until failure and assume that T is exponentially distributed with a failure rate α. Then the mean time to failure (MTTF) of a helicopter is $E(T) = 1/\alpha$. Similarly, the completion of repair on a machine is a death, so define S to be the service time of a repairman. The service time is also assumed to have an exponential distribution with a service rate μ. The mean time for repair (MTFR) is then $E(S) = 1/\mu$.

For these machine-repairman models, the state of the system, $N(t) = n$, is defined as the number of helicopters down at time t. Since there are only three helicopters in the models, then $n = 0,1,2,3$.

B. MODEL ESTIMATORS

In the study of queueing models, there are four quantities that are commonly used to evaluate the queueing system's performance. As part of the graphical output, the simulation will provide the user a table of theoretical and estimated values of these four quantities:

- L = the average number of machines in the repair system,
- L_q = the average number of machines in the repair queue,
- W = the average amount of time a machine is down,
- W_q = the average amount of time a machine waits for repair.

The cyclic nature of the model is clearly evident in the figure. A helicopter's life cycle consists of an operating period followed by, after a possible delay in queue, a repair period which is in turn followed by, after another possible delay in cold standby, another operating period.

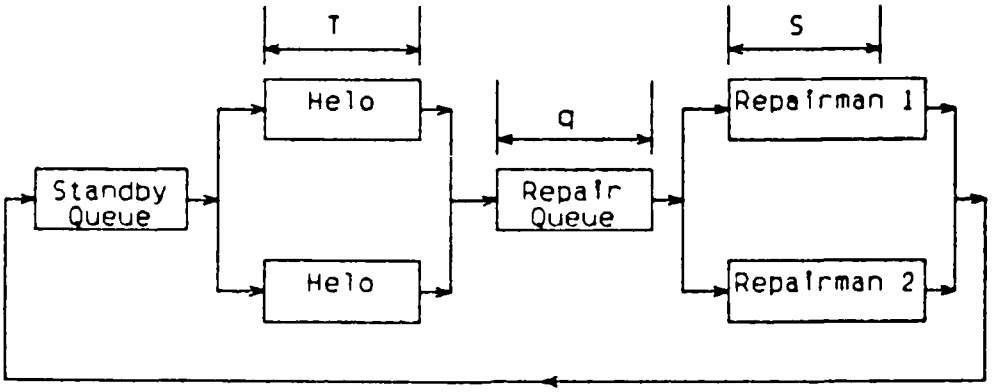


Figure 2: Machine-Repairman Model with Two Repairmen.

Figure 2 depicts a pictorial representation of the machine-repairman model with two repairmen [Ref. 3]. In this system, helicopters waiting in the repair queue go to the first available repairman for service. The maximum possible length of the repair queue is only one. As in the case of the one-repairman model, the standby queue can contain at most one helicopter.

The machine-repairman model is a special case of a birth-death process in which the event of a machine failing at time t constitutes a birth. Let T be the operating life

that is down and awaiting repair is colored red. When the down machine is in repair, it is seen with a repairman.

Figure 1 is a frequently used, pictorial representation (see [Ref. 2], for instance) of the machine-repairman model with one repairman. The dashed lines enclose the model's repair-queue system which contains a single-server queue. The machines outside the dashed box are the potential customers for the repair queue. Since there are only three machines in the model, the longest possible length of the queue will be two.

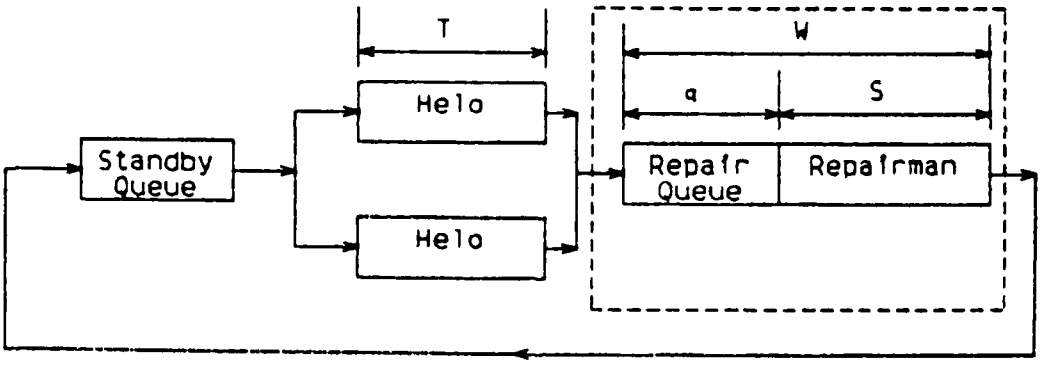


Figure 1: Machine-Repairman Model with One Repairman.

This model has a second queue, the cold-standby queue, since a helicopter may have to wait before being assigned to a mission. The standby queue can contain only one helicopter, and this occurs when there are no machines in the repair-queue system.

subject to failure while in cold standby since it is not operating.

A machine-repairman model typically assumes its machines operate continuously until failure. In the case of aircraft, this assumption does not reflect reality since in actual practice aircraft do not fly continuously, but have instead periods of inactivity between missions of limited duration. As already noted, one can reasonably assume that an inactive helicopter is not subject to failure and that the failure process occurs only when the helicopter is active.

The actual failure rate of a flying helicopter is the number of failures per flight hour. The product of the actual failure rate times the ratio of the number of flight hours performed to the number of calendar hours elapsed until failure defines the helicopter's effective failure rate. Thus the effective failure rate is the number of failures per calendar hour. To simplify the simulation, the models assume that the helicopters operate continuously until failure with the effective failure rate.

The selection of helicopters to represent the machines in these models allows a simple graphic design in depicting their status. An operating machine is seen in the display as a green, flying helicopter; a helicopter that is in cold standby is also colored green but is not flying. A machine

according to the current state and time values. As the time advances, the plotted lines will eventually reach the edge of the graph. When this condition is met, the clock module shifts the entire graph left one time unit and draws the new time increment. This procedure continues for all successive time increments. When the simulation clock has reached the time for the next event, the program jumps to either the failure module or the repair module as appropriate.

The failure module changes the helicopter display according to the current state and model parameters. The program keeps track of the machines individually, so that the graphic image of the particular, failing helicopter transforms from a green, flying figure to a red, grounded one. If a repairman is free, the graphic figure of a man is drawn next to the aircraft. If no repairman is available, then the human figure is omitted, signifying that the helicopter is placed in the repair queue. If a repairman was assigned, the failure module calls the random-number generator and computes the service time. The sum of the service time and the current time is the time that the repair will be completed and this is compared to the next-event times of the other machines after the program returns to the clock module.

In a similiar manner, the repair module changes the graphic figure of the helicopter that has just completed

repairs. If there are less than two helicopters flying when the repair event occurs, then the repaired machine transforms from red colored and disabled to green colored and flying. If two helicopters are already flying, then the repaired machine goes into the standby queue. The now free repairman moves to the helicopter at the head of the queue, or if the queue is empty, he disappears from the screen.

Before the program returns to the clock module, the repair module computes the interarrival time to failure of the helicopter just repaired. Failure times are not computed for helicopters placed in the standby queue. If the repairman is assigned to another helicopter, the repair module also computes the service time.

After returning to the clock module, the simulation updates the estimates of the tabulated quantities and prints them on the display screen. Both the failure and repair modules change the state-indicator variable to its new value so that the clock module will draw a horizontal line one time unit in length at the proper place on the graph.

SCRNSHFT is the subroutine that shifts the system state-versus-time graph one time unit to the left per clock cycle. The subroutine uses certain program transfers called interrupts to the PC-DOS Basic Input/Output System (BIOS) which contains routines that control the video

display. SCRNSHFT is not appreciably faster than a BASIC routine designed to perform the same task; however, the subroutine eliminates an annoying flicker in the display that was characteristic of the BASIC version.

D. PROGRAMMING CONSIDERATIONS

The program computes the interarrival (failure) and service times, assumed to be continuous, exponentially distributed, random variables, in the single-precision format. The current time, an integer value, is added to the interarrival time and to the service time to yield the next-failure time and the next-repair time respectively. In the unlikely event that the two single-precision, random variables match, the simulation will proceed as if the failure occurred earlier.

Whenever the model jumps from one state to another, the program computes a next-failure time and a next-repair time for the helicopter just affected by the transition. Thus MACHREPR always keeps in memory six next-event times for all machines regardless of their individual status. To determine the next event, the clock module compares the smallest of the three failure times with the smallest of the three repair times. With this method, there must be a means of preventing the program from simulating events considered impossible by the assumptions of the model.

For example, one model assumption is that a helicopter in cold standby is not subject to failure. It follows that the same machine cannot be repaired since it is not in the repair queue. Similarly, a helicopter that is in repair cannot fail and a helicopter that is flying cannot be repaired. To handle these impossible events, the program will set the failure time or the repair time, as appropriate, to equal 10^{31} which represents infinity.

Because of the discrete nature of the graphic display, all time values must be converted to integers prior to plotting the state-versus-time graph. It is for this reason that the random number representing the next-event time, be it a failure or a repair, is rounded up to the next higher integer. The result is that the interarrival and service times are not exponentially distributed but are instead geometrically distributed.

The discreteness of the time variable introduces a bias in the estimators since they are based on the assumption of continuous distributions. The effects of the bias, however, are minimal. The values of the estimates are still seen to converge to the limiting values as time advances. Despite the generation of geometric random variables, the simulation still provides a sufficient visual demonstration of the machine-repairman model.

E. RANDOM NUMBER GENERATION

The author found BASIC's random-number generator, a function called RND, unsuitable for simulation. Even so, a sample of 10,000 uniform variates produced by RND passed with .95 confidence three standard, nonparametric tests for uniformity and independence: the serial test, the frequency test, and the runs up-and-down test. On the other hand, a two-dimensional scatter plot of 5000 pairs of uniform variates, shown in Figure 4, clearly illustrate the lattice structure of the RND generator.

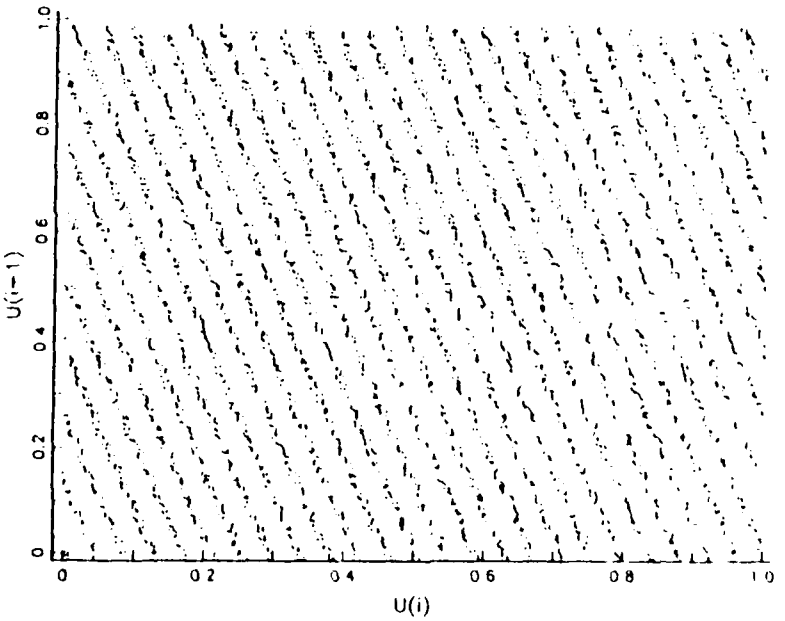


Figure 4: Two-Dimensional Plot of 5000 Pairs of Uniform Variates from the BASIC RND Function.

MACHREPR uses a random-number generator called RNGEN. Written in 8088 assembly language, RNGEN is modification of

a generator programmed by Associate Professor B.O. Shubert of the Naval Postgraduate School for use on the IBM PC. RNGEN uses the algorithm described by Fishman [Ref. 4] which is based on the expression

$$X_{i+1} = 16807 * X_i \bmod (2^{31} - 1)$$

where X_i is the old seed and X_{i+1} is the new seed. Dividing the new seed by the modulus produces the uniform (0,1), random number.

This algorithm is the one used by LLRANDOM, a random-number generator developed at the Naval Postgraduate School and described by Lewis, Goodman, and Miller [Ref. 5] as able to produce good quality, uniform variates.

The simulation computes exponential random numbers by the inverse probability integral transformation of uniform variates produced by RNGEN. Let U be a uniformly distributed, random number and let X be exponentially distributed with parameter α. An exponential random number is generated by setting $u = F(x)$ and solving for x,

$$u = F(x) = 1 - e^{-\alpha x}$$

$$1 - u = e^{-\alpha x}$$

$$\ln(1-u) = -\alpha x$$

$$x = -\ln(1-u)/\alpha, \text{ or}$$

$$x = -\ln(u)/\alpha.$$

LIST OF REFERENCES

1. Osgood, D., "A Computer on Every Desk," BYTE, v. 9, pp. 162-165, June 1984.

2. Allen, A.O., Probability, Statistics, and Queueing Theory, p. 187, Academic Press, 1978.

3. Ibid., p. 191.

4. Fishman, G.S., Concepts and Methods in Discrete Event Digital Simulation, pp. 179-180, Wiley, 1973.

5. Lewis, P.A.W., Goodman, A.S., and Miller, J.M., "A Pseudo-Random Number Generator for the System/360," IBM Systems Journal, v. 8, pp. 142-144, 1969.

APPENDIX A

THE MACHINE-REPAIRMAN MODEL USER'S GUIDE

by

Rex E. Nelsen

James D. Esary
Thesis Advisor

Alvin F. Andrus
Second Reader

TABLE OF CONTENTS

LIST OF FIGURES

A. INTRODUCTION

MACHREPR is a stochastic, discrete-event simulation of a machine-repairman model, which is a special case of birth-death processes. The model consists of three helicopters, of which two are in service and one is in cold standby, with an option of one or two repairmen. The program output is a graphics display composed of a system state-versus-time graph, a table of statistics, and animated figures that illustrate the dynamics of the process.

This manual discusses the model and guides the user in the step-by-step operation of the program. In describing commands and keyboard entries, the following notation is used. Single quotes (' ') enclose commands and phrases that are to be entered verbatim. The act of pressing a specific key is indicated by brackets (< >) encasing the letter or letters on the key. For example, 'format b:' <ENTER> means type the command exactly as it appears between the quotes and follow it by pressing the enter key. This user's guide shows the commands in the lower case; however, the IBM PC will accept upper case letters as well.

B. MODEL DESCRIPTION

MACHREPR contains two versions of the machine-repairman model: the one-repairman case and the two-repairman case. In both cases, the nominal mission of the model is to keep two helicopters flying at all times. If the third

helicopter is serviceable, it is kept in cold standby as an immediate replacement for the other two. Upon failure of a helicopter, a repairman is assigned if he is available; otherwise the helicopter joins a queue to await repair.

Disabled (down) helicopters remain in the repair queue until a repairman becomes free. The repair queue is a first in, first out type (FIFO). A newly repaired helicopter will go immediately into flying operation if less than two aircraft are in service. If there are already two helicopters flying when a repair is completed, then the third goes into the cold-standy queue.

There are several simplifying assumptions for these models. All helicopters are assumed to have exponentially distributed service lives with the same failure rate α. If T is defined as the length of a helicopter's operating life, then the mean time to failure (MTTF) is $E(T) = 1/\alpha$. The simulation uses the MTTF as the input parameter that establishes the particular exponential distribution for the operating life. Helicopters will fly continuously until failure. A machine that is in cold standby is assumed not to be subject to failure.

Another assumption is that the service time for the repairmen has an exponential distribution with parameter μ. If S is the time required to complete a repair job, then the mean time for repair is $E(S) = 1/\mu$. Like the MTTF, the mean time for repair (MTFR) is an input variable for the

simulation. The repairmen are assumed to be equally competent and thus have the same MTFR.

In both cases of the machine-repairman model, the state of the system is defined as the number of helicopters down at time t. The state variable is $N(t) = n$, $n = 0,1,2,3$.

C. HARDWARE AND SOFTWARE

1. Hardware

The simulation is programmed for the IBM Personal Computer (IBM PC) with a color monitor. Since the program uses color graphics, the microcomputer must be equipped with IBM's Color/Graphics Adapter or a suitable graphics card from another manufacturer. MACHREPR will not run without the graphics adapter. The program uses two assembly-language subroutines that are loaded in memory outside of BASIC's 64 kilobyte (K) workspace; therefore, there must be a minimum of 128 K of memory installed in the microcomputer. The computer must have at least one 5¼-inch, floppy disk-drive.

2. Software

MACHREPR and the two subroutines are supplied on a distribution diskette. The user must provide the disk operating system PC-DOS 2.0 (or 2.10) and Advanced BASIC (BASICA) in order to run the simulation. MACHREPR uses BASIC's POKE command and certain program transfers, called interrupts, to the Basic Input/Output System (BIOS) in the

IBM ROM (read-only memory). Because of these interrupts, there can be no assurance that the simulation will operate properly on other "IBM compatable" machines.

3. Program Files

The distribution diskette contains the following program files:

* MACHREPR.BAS - the main program.
* RNGEN.SRT - a random-number generator written in 8088 assembly-language used as a subroutine to the main program.
* SCRNSHFT.SRT - another assembly-language subroutine used to perform a specific task on the graphics screen.
* RNGEN.LST - a source-code listing.
* SCRNSHFT.LST - a source-code listing.
* MACHREPR.BAT - a batch file that loads and starts the simulation.

The simulation requires the first three files listed above in order to run. The two subroutines are binary-image files and can not be executed from PC-DOS. The two source-code listings are provided as a convenience to the user. MACHREPR.BAT is a simple example of how a batch file can be used to easily start the simulation.

D. GETTING STARTED

1. Making a Backup Copy

The first step is to make a copy of the distribution diskette. After the copy is made, the original

graph. With exponentially distributed interarrival and service times, there is a non-zero probability that the sojourn in any particular state will be less than one time unit. The program will still alter the helicopter display properly and compute the estimates correctly; however, the state-versus-time graph may display a jump of two states because of the program's inability to plot a line of less than one time unit in length. The number of times this discrepancy occurs can be minimized if the MTTF and the MTFR are set at values greater than ten.

a result, the estimators are biased since they are based on the assumption of continuous time when in fact they are calculated with data based on discrete increments of time. The effects of the bias are minimal; the simulation still provides a sufficient demonstration of the convergence of the estimates to the limiting values.

G. KEYBOARD COMMANDS

There are three keyboard commands available to the user as the simulation is in progress. When <p> (for pause) is pressed, the simulation freezes to give the user the opportunity to study the graphic display. To continue the simulation, press <c>. The third command is <s> for stop the simulation. The <s> key will send the program back to the program menu.

H. PROGRAM LIMITATIONS

MACHREPR always starts a simulation under the same conditions. The initial state is N(0) = 0 with helicopters B and C in flight and helicopter A in cold standby. There are no provisions in the program for starting the simulation in another state. Restarting the simulation from the program menu causes the current time and all data collection variables to be set to zero.

As noted before, all event times are rounded up to the next higher integer for plotting the state-versus-time

* Lq - The average number of machines in the repair queue.

* W - The average amount of time that a machine is in the repair system. This quantity is the sum of Wq and the expected service time.

* L - The average number of machines in the repair system. This quantity includes those machines in the repair queue and those machines in repair.

* Avl- The machine availability defined as the proportion of the time that at least one helicopter is flyable.

* Pd - The probability that a down machine must wait for repair.

* Ps - The probability that an up machine must wait in cold standby.

The numbers that appear in the right-hand column under the heading "Limit" are the limiting values of the quantities of interest. In other words, the quantities converge to the limiting values as time goes to infinity. The limiting values are calculated with the balance equations that correspond with the particular machine-repairman model.

The values in the left-hand column are estimates of the limiting values. These quantities are computed with data collected from the simulation. Upon the occurance of an event, the program recomputes the estimates after taking into account the newly collected data from the last state.

Because the plots on the state-versus-time graph must be given in terms of integer coordinates, all time variables must be rounded up to the next higher integer. As

running the two-repairman model, then the program will draw the second repairman beside the helicopter that fails after the one already in repair.

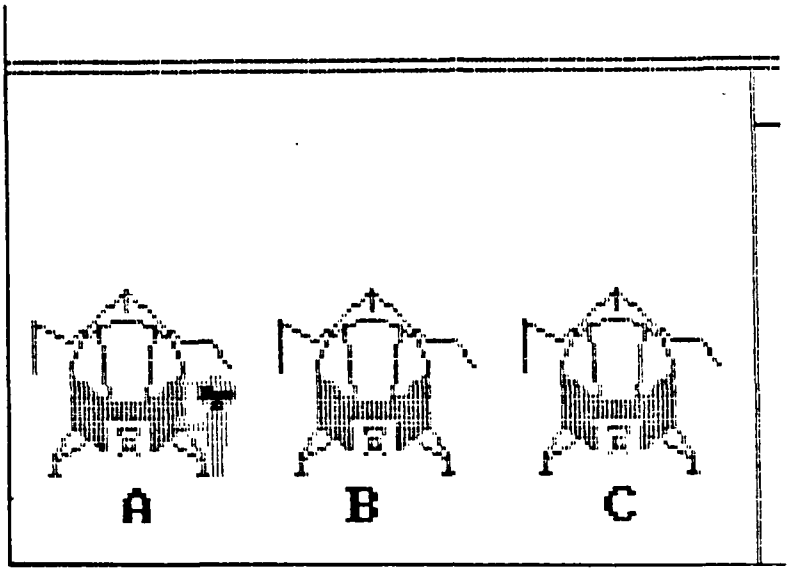


Figure 11: Helicopter Display in State 3.

3. The Quantities of Interest

In the study of queueing systems, there are several quantities of interest that are used to evaluate the system's performance. In the lower right corner of the graphics display (see Figure 7) there is a table of the more commonly used quantities.

The row labels have the following definitions:

* AR - The average arrival rate of machines into the repair system.

* Wq - The average waiting time of a machine in the repair queue.

and service (repair) times. The program changes the appropriate graphic figures when the simulation clock reaches the time for the next event.

Figure 10 shows an example of a helicopter display that indicates state 2. In this case, C is in repair. Helicopter A is also disabled, and since it does not have a repairman, A is in the repair queue.

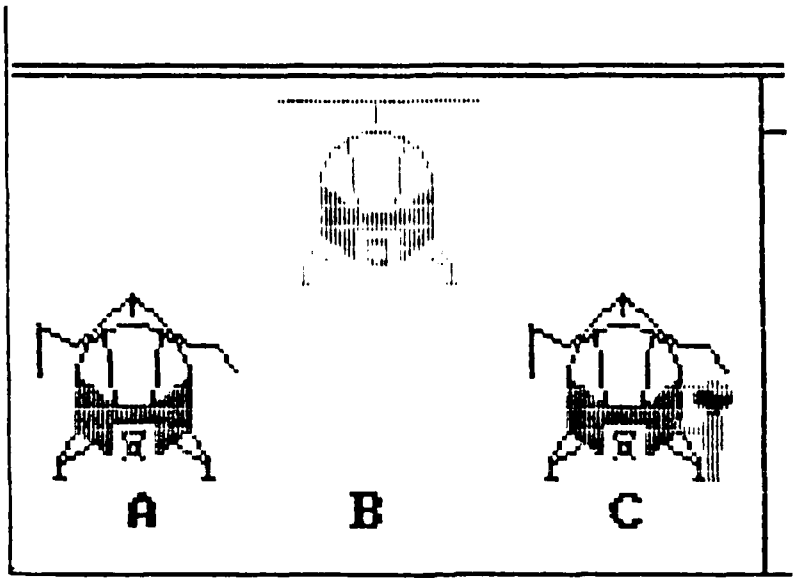


Figure 10: Helicopter Display in State 2.

In Figure 11, all three machines are down, thus the system is in state 3. There is a repair queue of length two in this example.

It should be noted that Figures 10 and 11 are illustrations from the one-repairman model since only one repairman appears in either display. If the simulation is

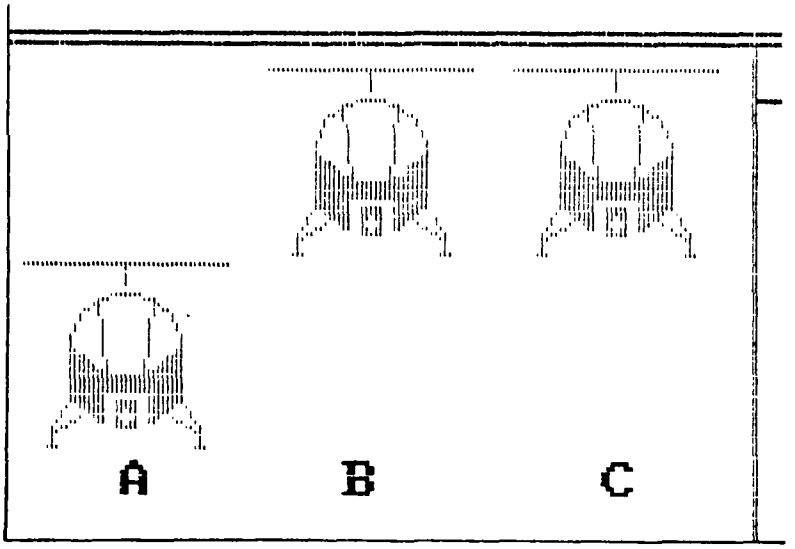


Figure 8: Initial Helicopter Display in State 0.

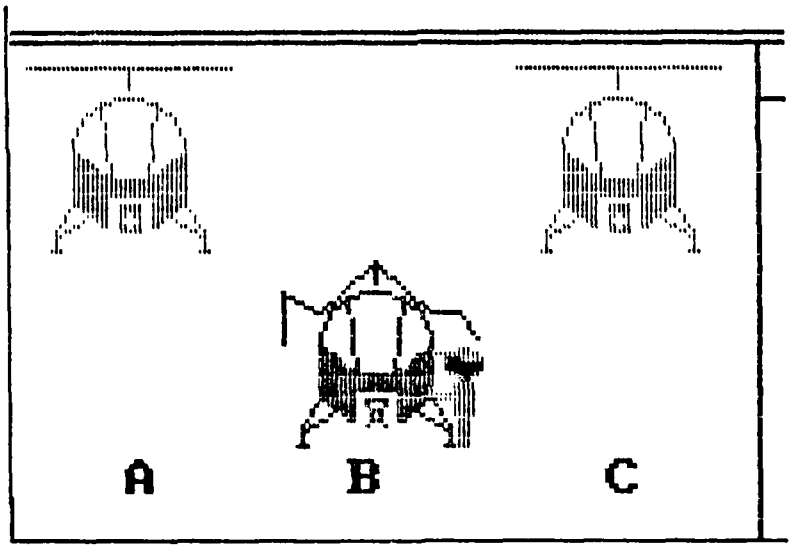


Figure 9: Helicopter Display in State 1.

As the simulation proceeds, random numbers are generated and transformed into interarrival (failure) times

at the height corresponding to the current state of the system. The accumulation of these individual plots forms a step graph as shown by the example in Figure 7. When the graph reaches the end of the display, the program makes room for the next plot by calling SCRNSHFT to shift the entire graph one unit to the left. In this way, the model's activities for the past fifty time units can be seen in a single glance.

2. The Helicopter Display

The three helicopter images in the lower left corner of the graphic display indicate the current state of the system. The display in Figure 8 shows how the helicopters are arranged in the initial state 0. All three machines are colored green. Helicopters B and C, seen in the upper half of the display, are flying. Helicopter A on the "ground" is in cold standby.

Since all helicopters are initially either flying or in cold standby, the next event will be a failure in either machine B or C. A machine that fails is grounded and transformed into a red colored helicopter with a broken rotor. Because there is only one down helicopter, the system is in state 1 and so there is at least one idle repairman. The program draws a graphic figure of a man beside the down aircraft to represent a machine that is in repair. For example, Figure 9 depicts state 1 with helicopter B in repair.

graphics display is divided into three sections: the state-versus-time graph, the helicopter display, and the table of quantities used with queueing models.

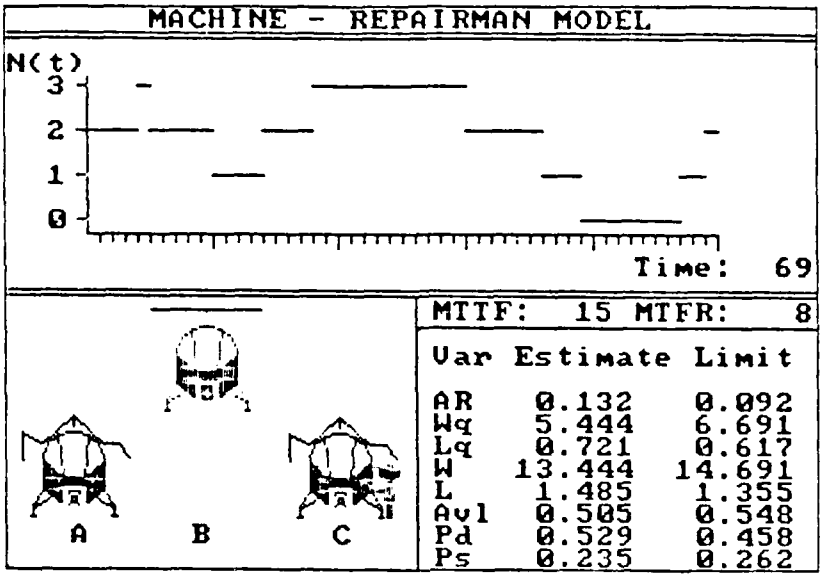


Figure 7: An Example of the Graphics Display.

1. The State-Verses-Time Graph

The upper half of the display contains the state-versus-time graph. The values of the state variable N(t), the number of machines down at time t, are on the ordinate. There are fifty divisions on the abscissa which represent units of time. The current time of the system is printed at the right-hand end of the abscissa.

With each unit advancement of the simulation clock, the program plots a horizontal line one time unit in length

and function properly with non-integer values of MTTF and MTFR; however, these values will be printed on the graphics display as integers. Negative numbers will not be accepted for the MTTF and the MTFR. The digits 1 and 2 are the only valid responses for the number of repairmen. The program will immediately start the simulation after the number of repairmen has been entered.

3. The Default Parameters

If this option is selected, then the simulation starts with the model parameters set at default values. These values are:

MTTF = 15 time units,

MTFR = 10 time units,

Number of repairmen = 1.

The default parameters are selected by pressing <d>.

4. Ending the Program

To end the program, press <e>. MACHREPR will clear the screen, exit the graphics mode, and leave the computer in the BASIC command mode. If the user wants to exit BASIC and return to the DOS, type 'system' <ENTER>.

F. THE GRAPHICS DISPLAY

MACHREPR is designed to give the observer the illusion of real-time passage as the probabilistic events occur in the machine-repairman system. The graphics display indicates the model's activities. As shown in Figure 7, the

1. Setting the Random-Number Generator Seed

To set the seed, press the <s> key. The program will then respond by asking for the value of the seed. The seed is entered by typing any integer in the range of 1 to 2,147,483,646 ($= 2^{31} - 2$) and pressing <ENTER>. The program will accept a non-integer value; however, during processing the number will be truncated to an integer, with the result possibly being different than the desired value. After the seed has been entered, the program returns to the program menu.

It is not necessary to set the random-number generator seed to run the simulation. There is a default seed embedded in the generator's program code, and it is updated automatically with each call for a random number. If the simulation is stopped and then restarted without setting the seed, the generator will use as the initial seed the value remaining from the last call in the last simulation run.

2. Changing the Model Parameters

When <c> is pressed, MACHREPR will successively ask for the desired values of the mean time to failure, mean time for repair, and the number of repairmen. The quantities are entered by typing a positive integer and pressing <ENTER> in response to each question. For best results, the values for the mean time to failure and the mean time for repair should be at least ten. The program will accept

parameters that can be altered by the user are the helicopter's mean time to failure, the repairmen's mean time for repair, and the number of repairmen to employ. Additionally, the initial seed for the random-number generator can be set.

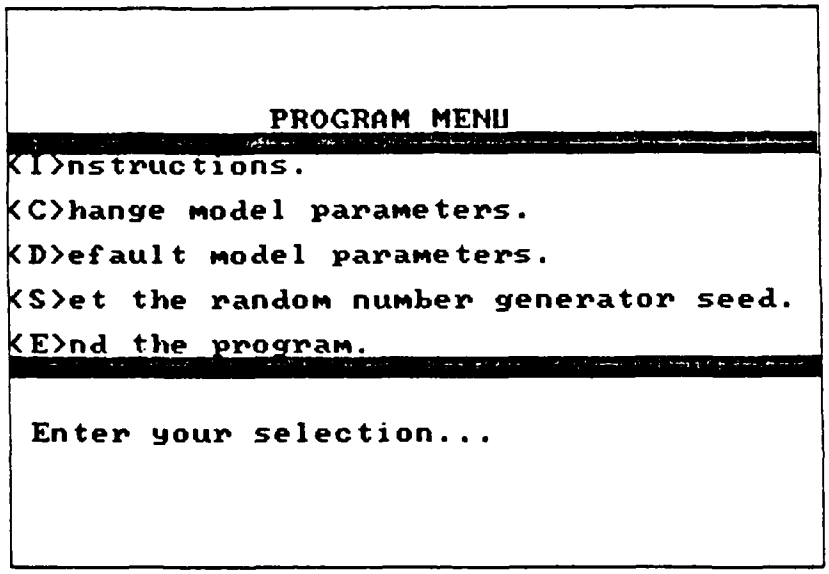


Figure 6: The Program Menu.

The simulation starts and ends with the program menu. Whenever the user stops the simulation or whenever the simulation clock reaches the preset limit of 9950 time units, the program returns to the menu. The simulation then can be restarted with perhaps a difference set of model parameters, or the program can be ended altogether.

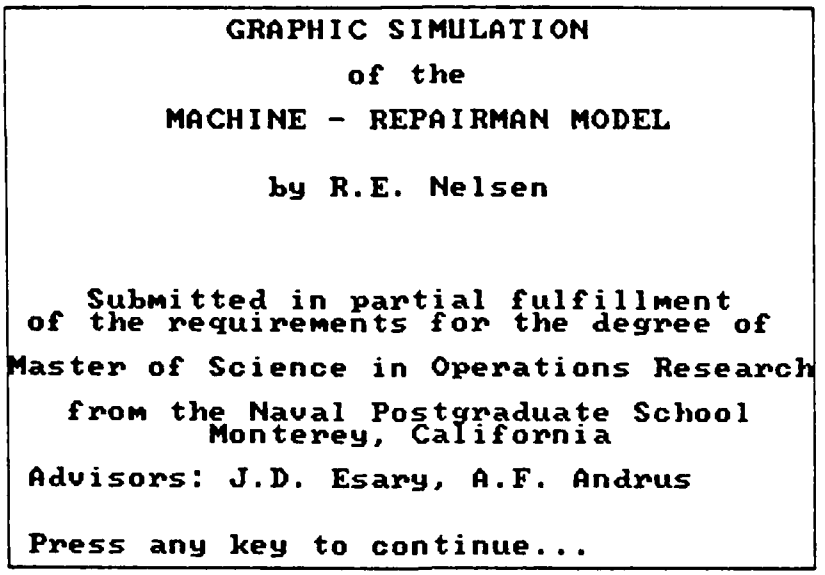


Figure 5: The Title Screen.

3. Title Screen

The display will remain dark for a few seconds while the program initializes variables, arrays, and function definitions. When initialization is done, MACHREPR will present the title screen as shown in Figure 5.

E. THE PROGRAM MENU

The program menu, depicted in Figure 6, appears next and provides the means of controlling the simulation. With the options available on the menu, the user may choose to read the program instructions, change the model parameters or use the default model parameters. The program instructions that are available on the screen are a summarized version of those found in this manual. The model

command to make an identical copy of the distribution diskette.

2. Starting the Simulation

If the first option was used, insert the bootable diskette into drive A and turn on the microcomputer. After a few seconds, the DOS prompt A> will appear on the screen. It is here that the batch file MACHREPR.BAT can be used. Simply type 'machrepr' <ENTER> to start the simulation.

If using the non-bootable diskette, then turn on the microcomputer with the DOS diskette in drive A. Next load Advanced BASIC into memory by typing 'basica' <ENTER>. After the computer is in BASIC's command mode, remove the DOS diskette from drive A and insert the diskette containing the simulation. Type 'load "machrepr.bas",r' <ENTER> to bring the program into memory and to start the simulation.

If desired, the simulation can be started from a second disk-drive. Begin as before by turning on the microcomputer with the DOS diskette in drive A and loading BASICA. With the application diskette inserted into drive B, type 'load "b:machrepr.bas",r' <ENTER>. During execution, MACHREPR reads the program diskette and expects to find it in the default disk-drive. When the program fails to find the required files on the DOS diskette in drive A, it asks the operator for the correct drive label. Simply respond to the program's question by pressing <b>.

diskette should be stored in a safe place and used only to make a replacement for an unserviceable application diskette. Although there are several ways that the user may choose to create an application diskette, this manual will suggest just two:

* A self-starting (bootable) diskette containing the operating system files, the BASICA command file, and the simulation's program files.

* A simple backup copy which contains only the program files and which must be used in conjunction with a system diskette.

Owners of microcomputers with a single disk-drive may find the first option more convenient.

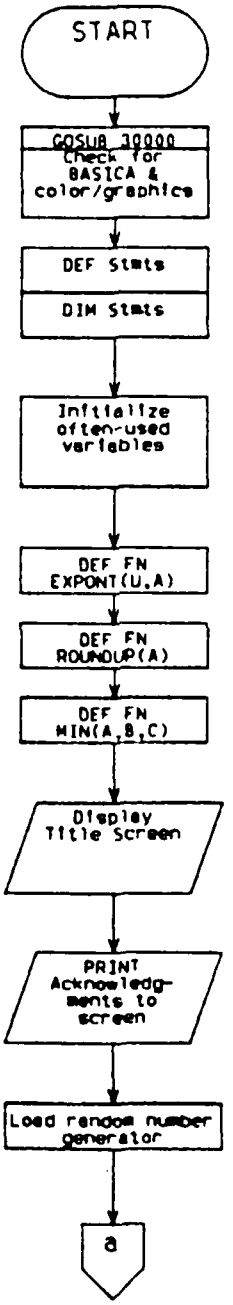
To make a bootable diskette, follow the instructions in the PC-DOS manual and format a blank diskette using the FORMAT command with the /S parameter. The /S parameter causes the computer to transfer the operating system files from the DOS diskette to the newly formatted diskette. The Advanced BASIC file, called BASICA.COM, is also found on the DOS diskette and can be copied onto the application diskette with the COPY command. Finally, use the COPY command again to transfer the simulation's six program files from the distribution diskette onto the bootable diskette.

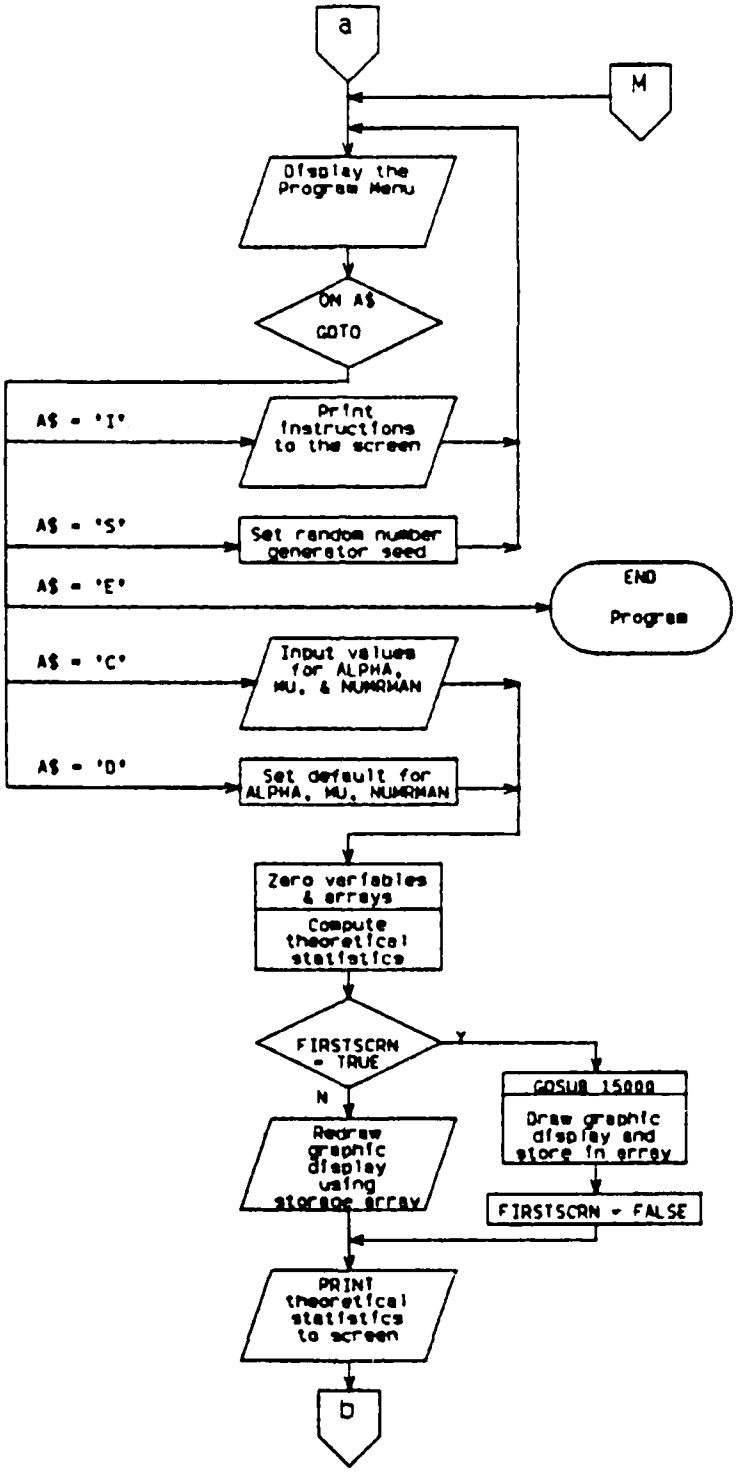
If the second option is desired, then format a blank diskette without the /S parameter. This procedure does not transfer the operating system files to the application diskette. The next step is to use the DISKCOPY

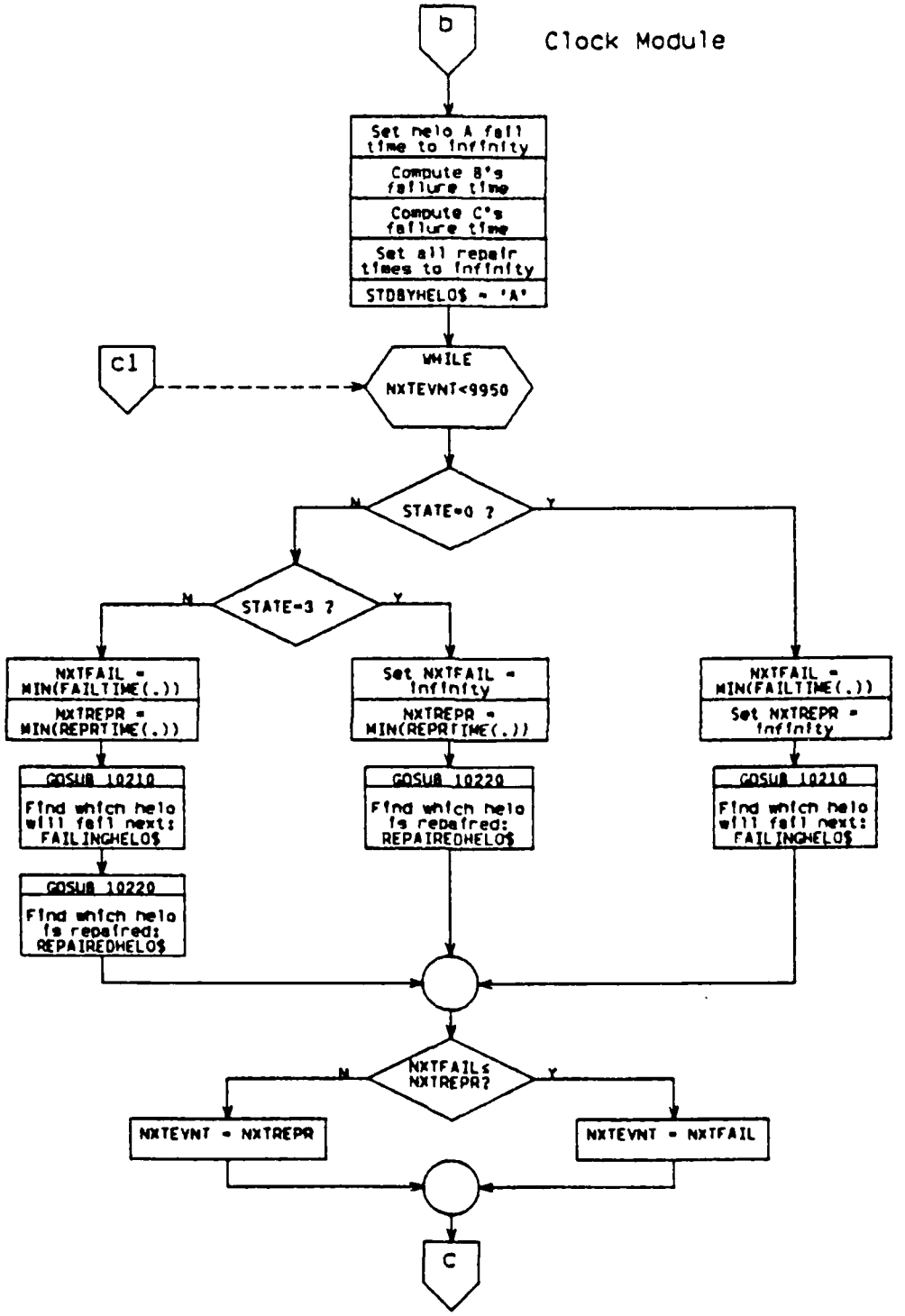
APPENDIX B

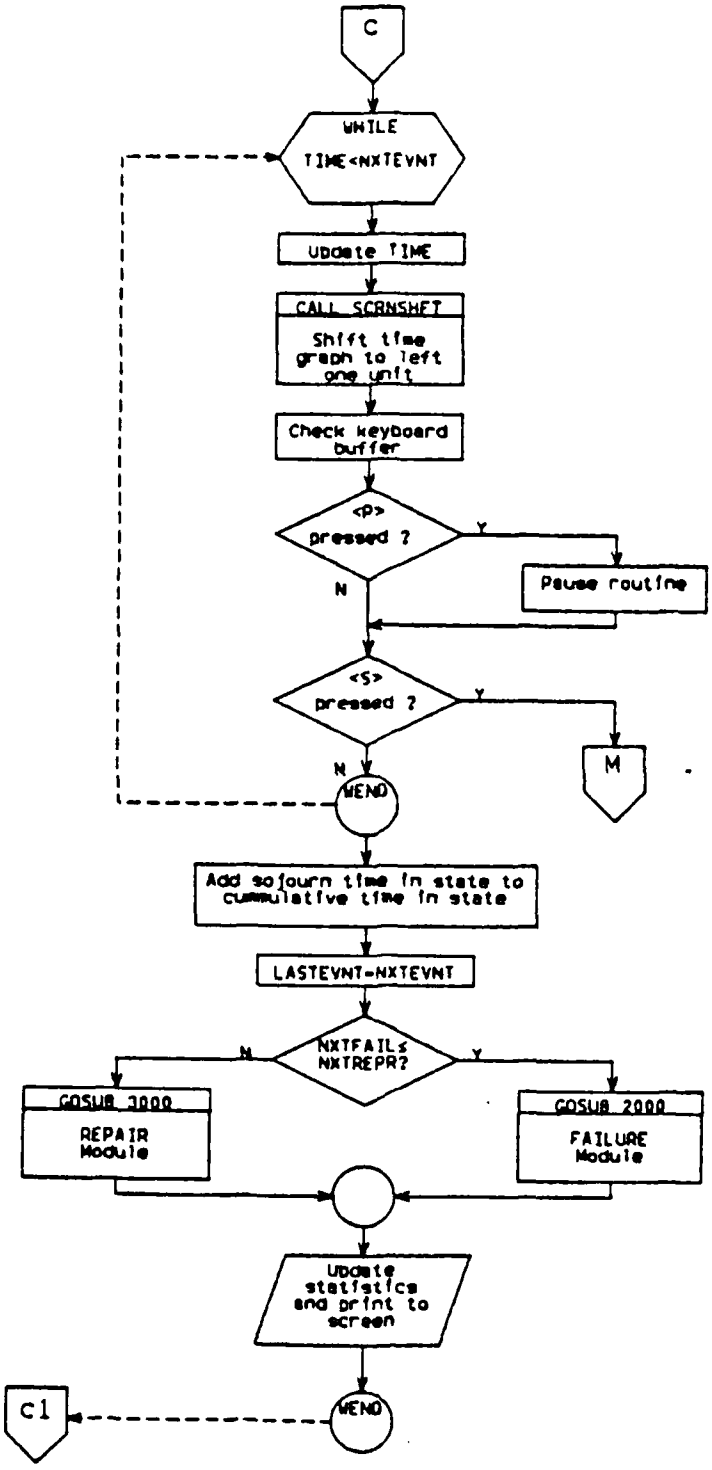
MACHREPR.BAS FLOWCHART

Main Program

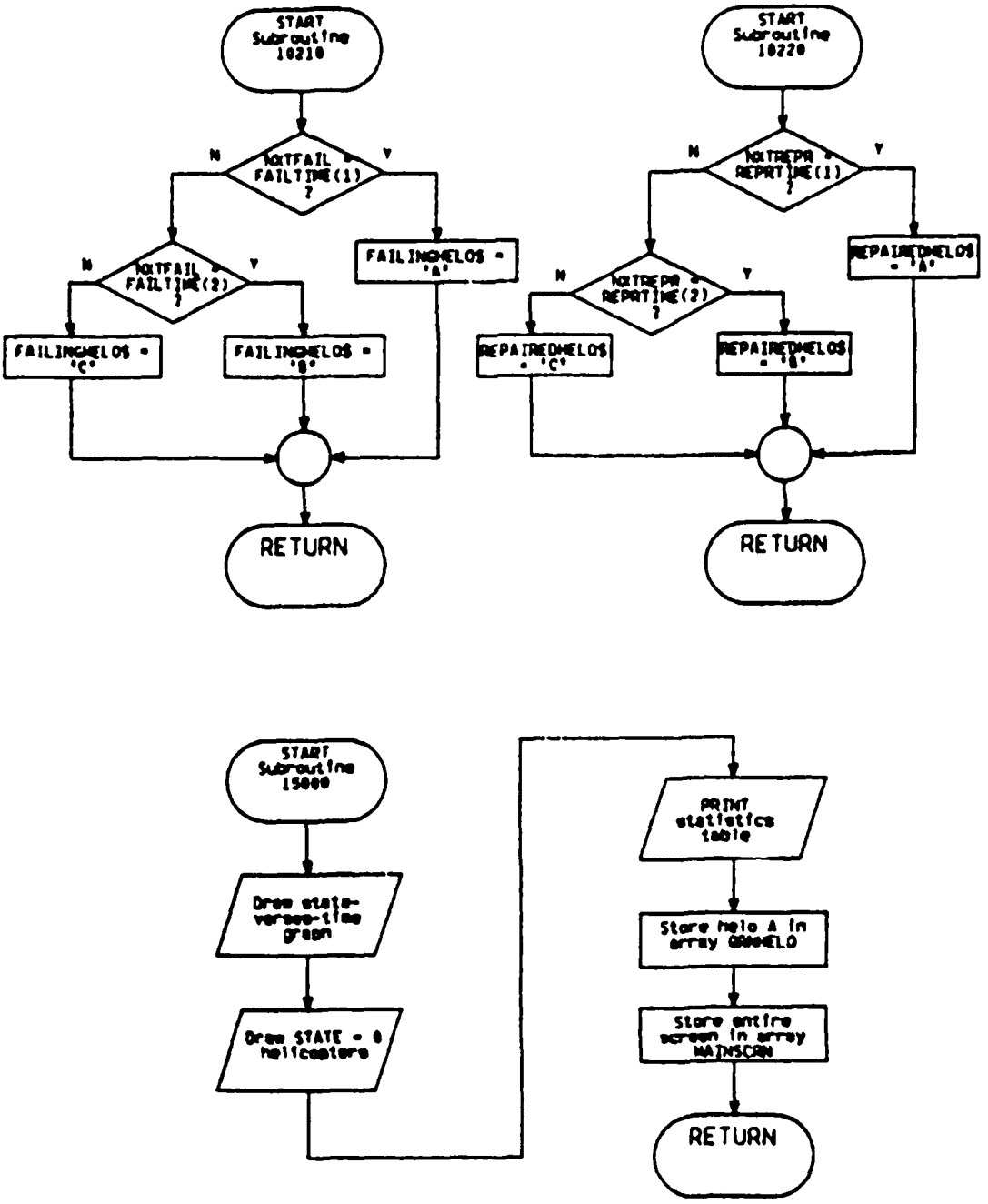



a
M
Display the Program Menu
ON A$ GOTO
A$ = 'I'
Print instructions to the screen
A$ = 'S'
Set random number generator seed
A$ = 'E'
END Program
A$ = 'C'
Input values for ALPHA, MU, & NUMRMAN
A$ = 'D'
Set default for ALPHA, MU, NUMRMAN
Zero verfables & arrays
Compute theoretical statistics
FIRSTSCRN = TRUE
Y
N
GOSUB 15000
Draw graphic display and store in array
FIRSTSCRN = FALSE
Redraw graphic display using storage array
PRINT theoretical statistics to screen
b

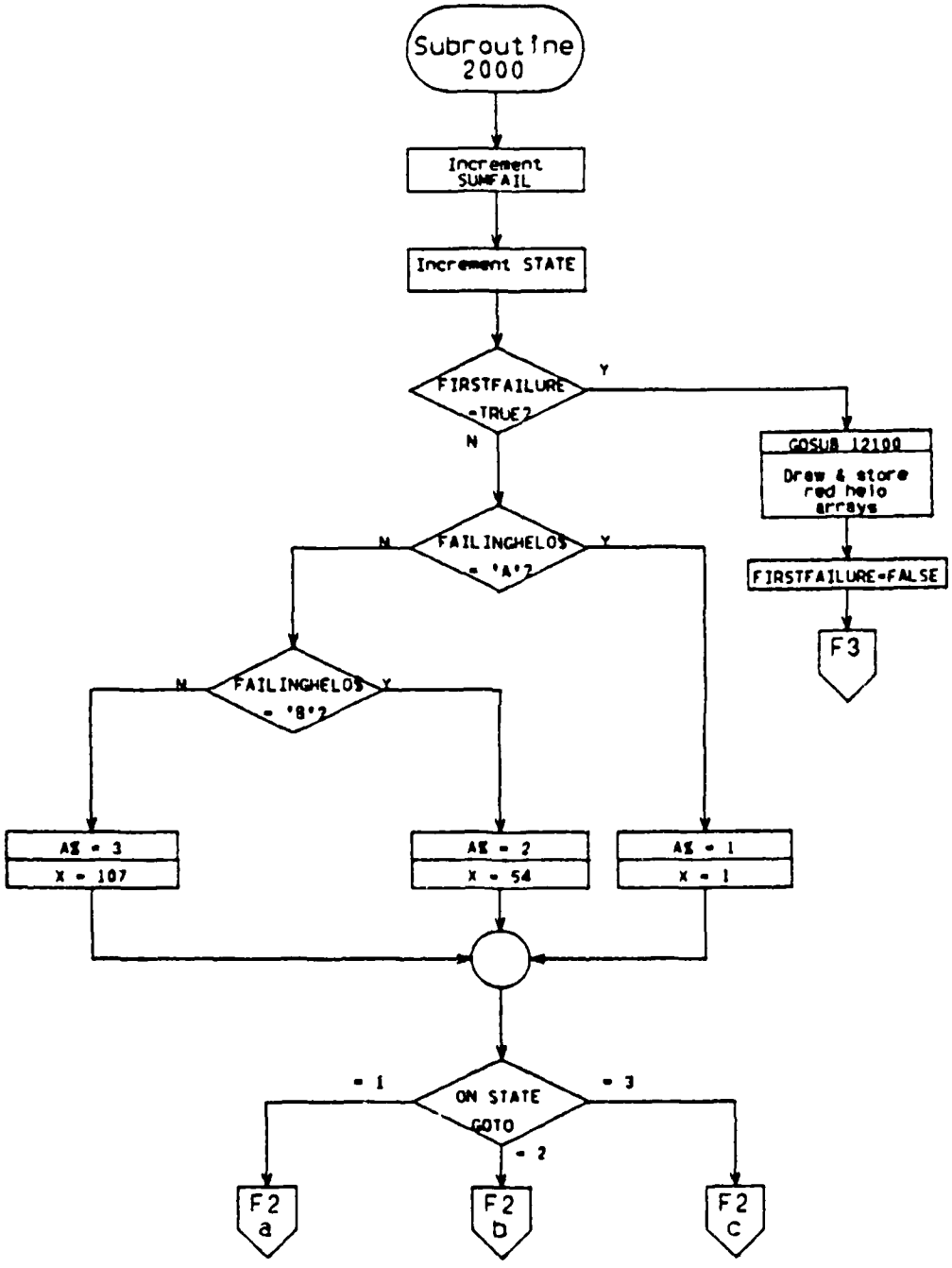

b
Clock Module
Set helo A fail time to infinity
Compute B's failure time
Compute C's failure time
Set all repair times to infinity
STDBYHELO$ = 'A'
C1
WHILE NXTEVNT<9950
STATE=0 ?
N
Y
STATE=3 ?
N
Y
NXTFAIL = MIN(FAILTIME(.))
NXTREPR = MIN(REPRTIME(.))
GOSUB 10210
Find which helo will fail next: FAILINGHELO$
GOSUB 10220
Find which helo is repaired: REPAIREDHELO$
Set NXTFAIL = infinity
NXTREPR = MIN(REPRTIME(.))
GOSUB 10220
Find which helo is repaired: REPAIREDHELO$
NXTFAIL = MIN(FAILTIME(.))
Set NXTREPR = infinity
GOSUB 10210
Find which helo will fail next: FAILINGHELO$
NXTFAIL≤ NXTREPR?
N
Y
NXTEVNT = NXTREPR
NXTEVNT = NXTFAIL
C


C
WHILE
TIME<NXTEVNT
Update TIME
CALL SCRNSHFT
Shift time
graph to left
one unit
Check keyboard
buffer
<P>
pressed ?
Y
Pause routine
N
<S>
pressed ?
Y
M
N
WEND
Add sojourn time in state to
cumulative time in state
LASTEVNT=NXTEVNT
NXTFAILS
NXTREPR?
N
Y
GOSUB 3000
REPAIR
Module
GOSUB 2000
FAILURE
Module
Update
statistics
and print to
screen
WEND
C1

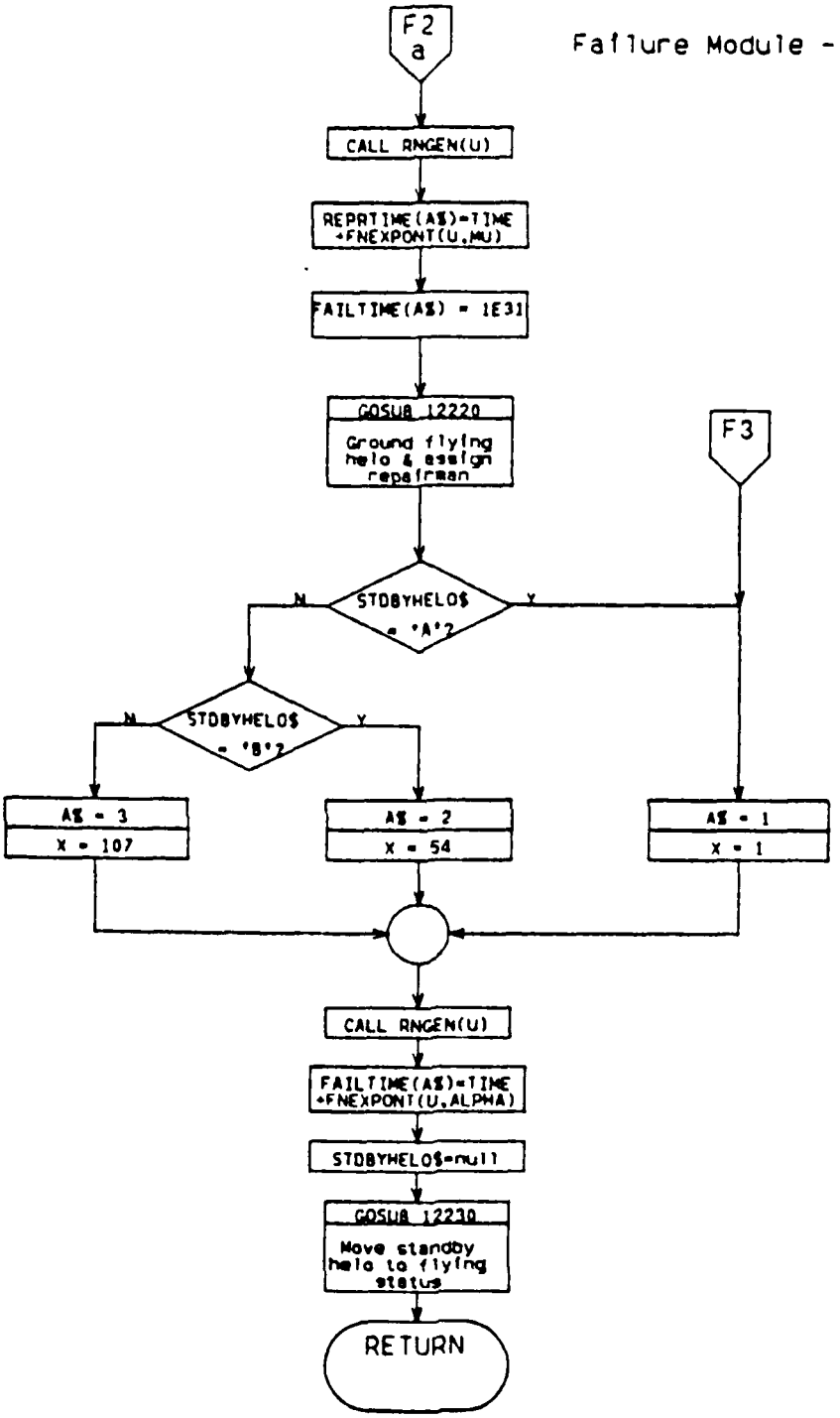
Clock Module Subroutines

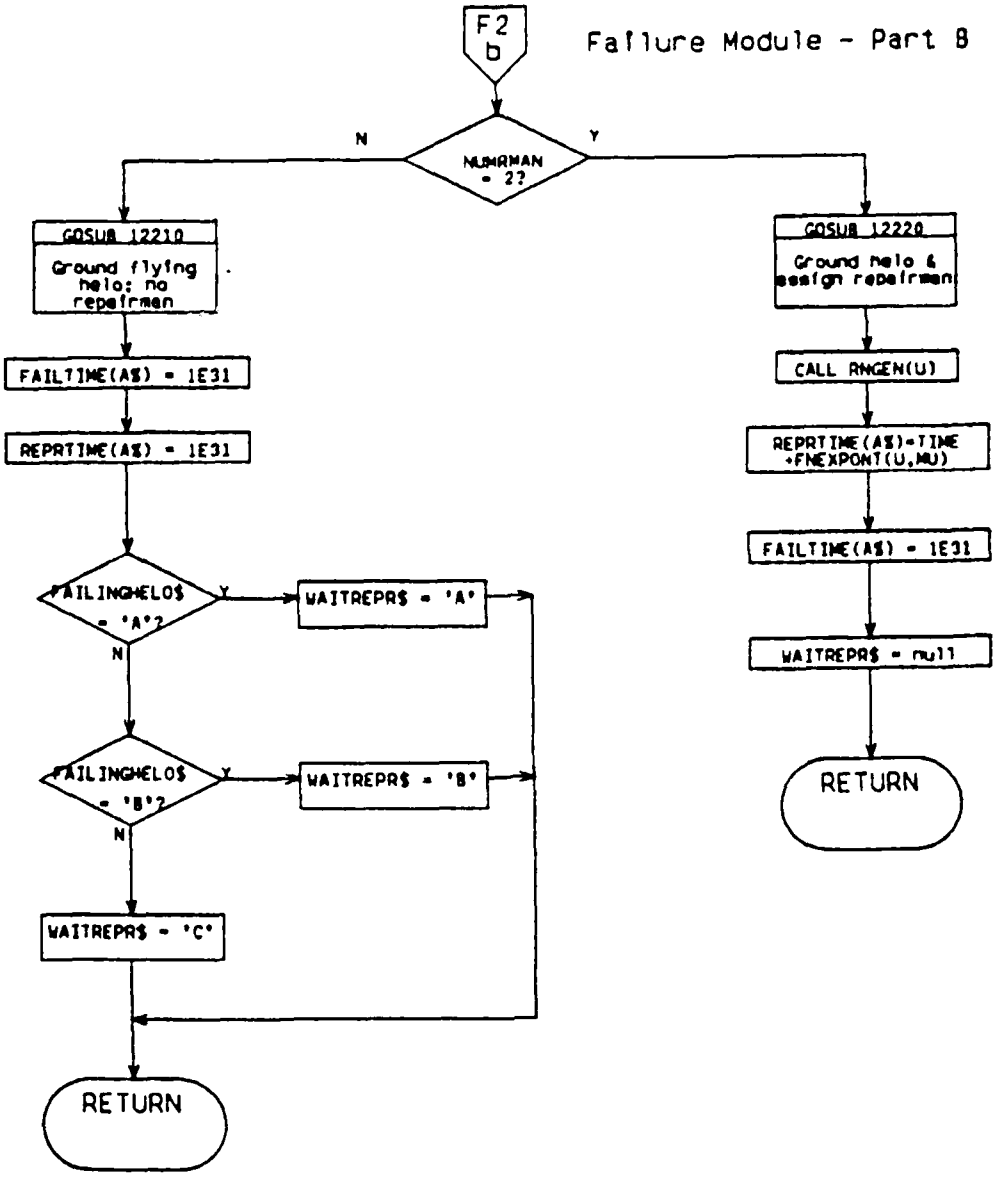


Failure Module

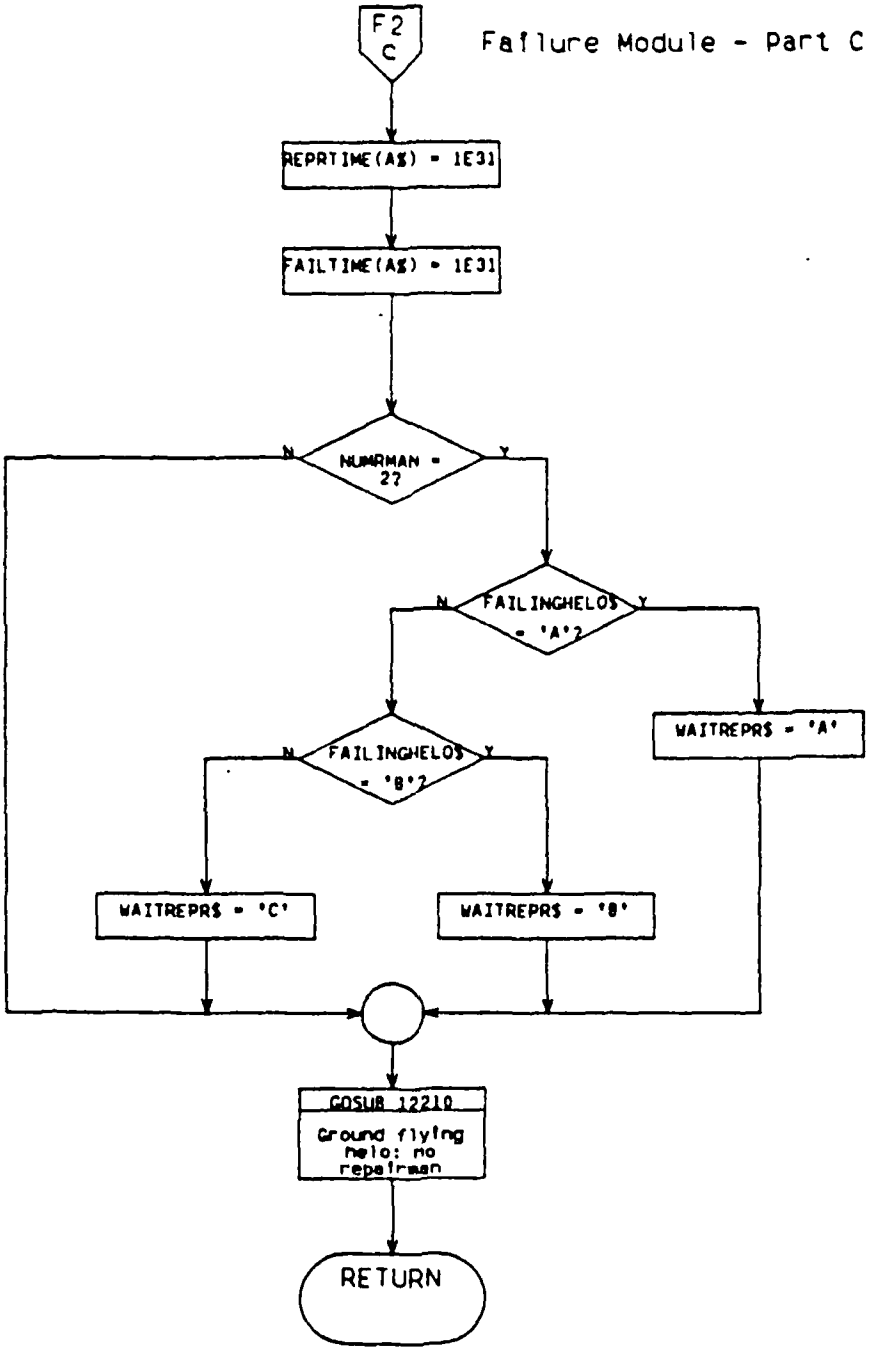


Failure Module - Part A

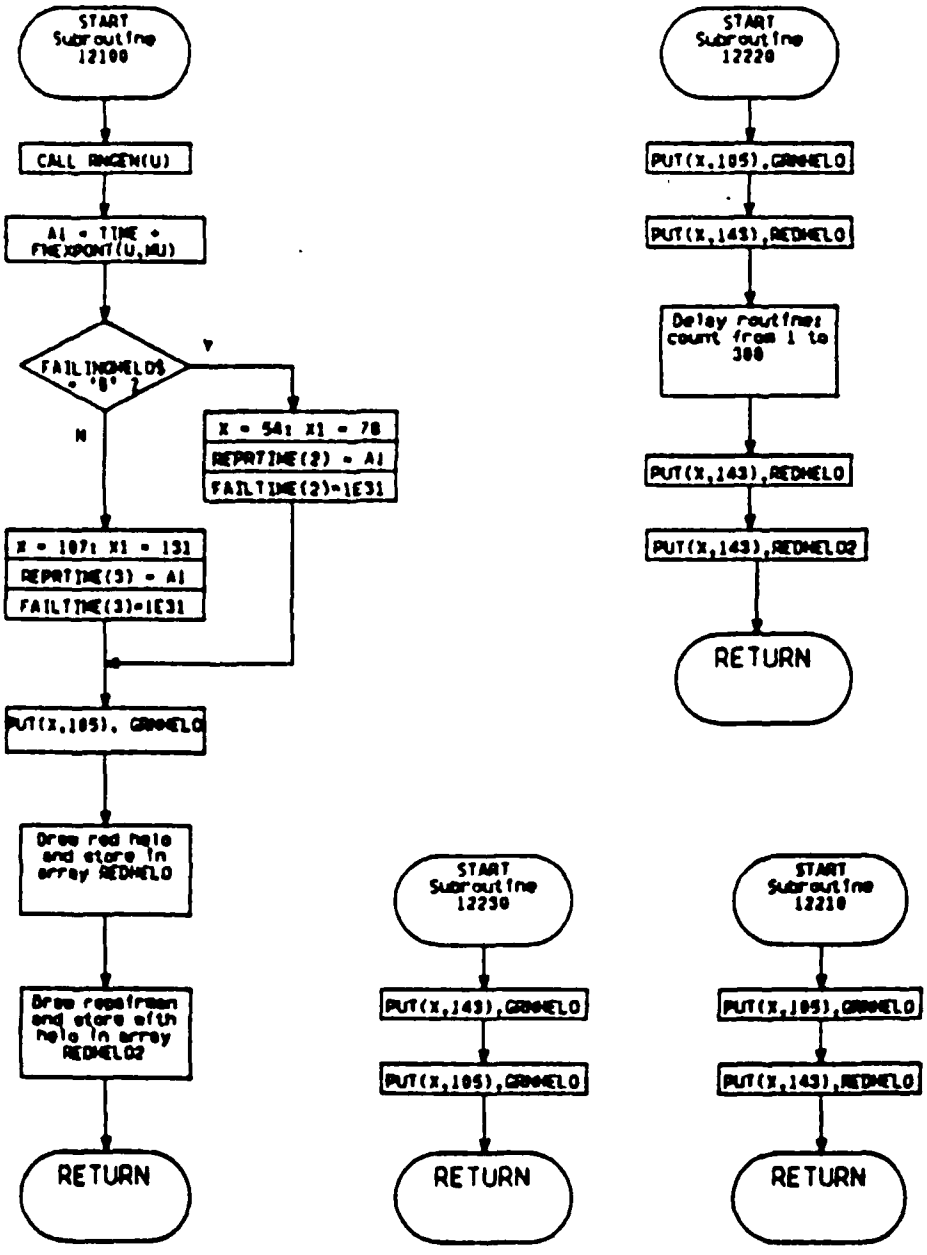



F2
b
Failure Module - Part B
NUMAMAN
= 2?
N
Y
GOSUB 12210
Ground flying
helo: no
repairmen
FAILTIME(A%) = 1E31
REPRTIME(A%) = 1E31
FAILINGHELO$
= 'A'?
Y
N
WAITREPR$ = 'A'
FAILINGHELO$
= 'B'?
Y
N
WAITREPR$ = 'B'
WAITREPR$ = 'C'
RETURN
GOSUB 12220
Ground helo &
assign repairmen
CALL RNGEN(U)
REPRTIME(A%)=TIME
+FNEXPONT(U,MU)
FAILTIME(A%) = 1E31
WAITREPR$ = null
RETURN

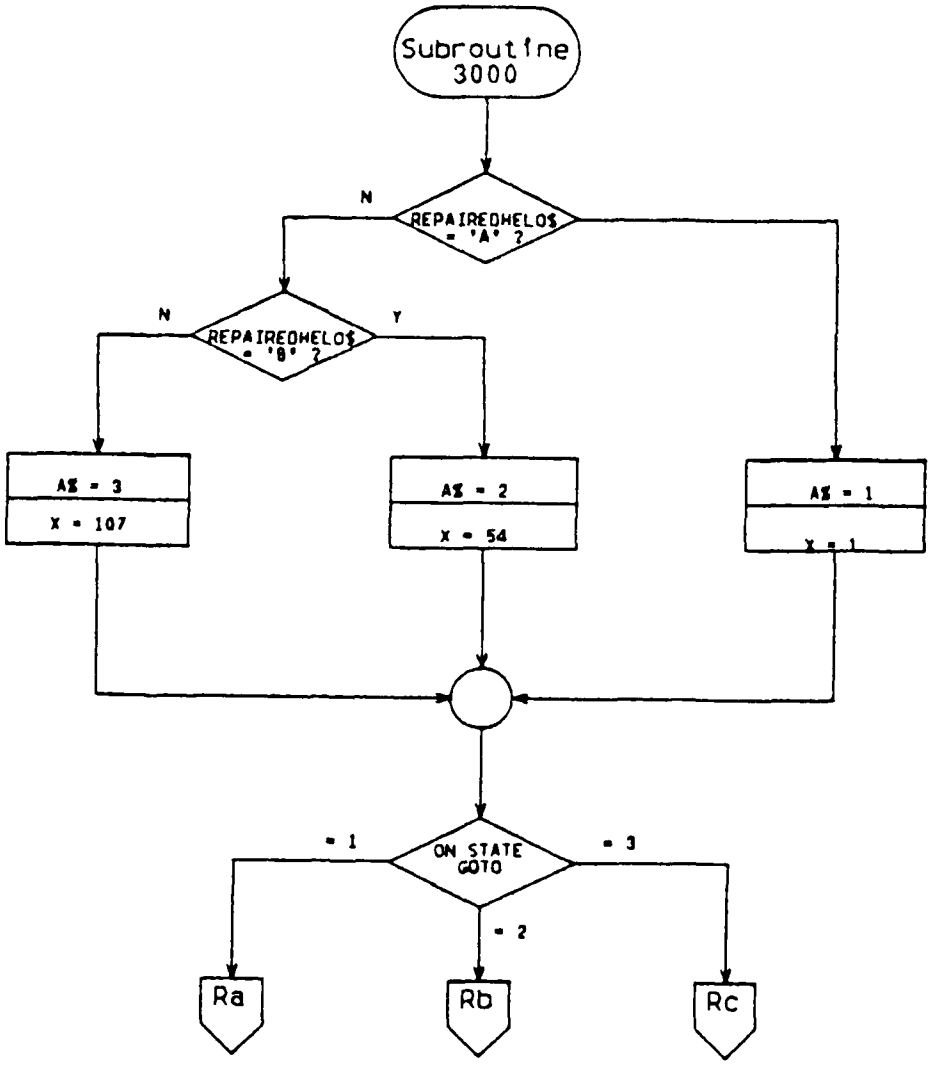
Failure Module - Part C

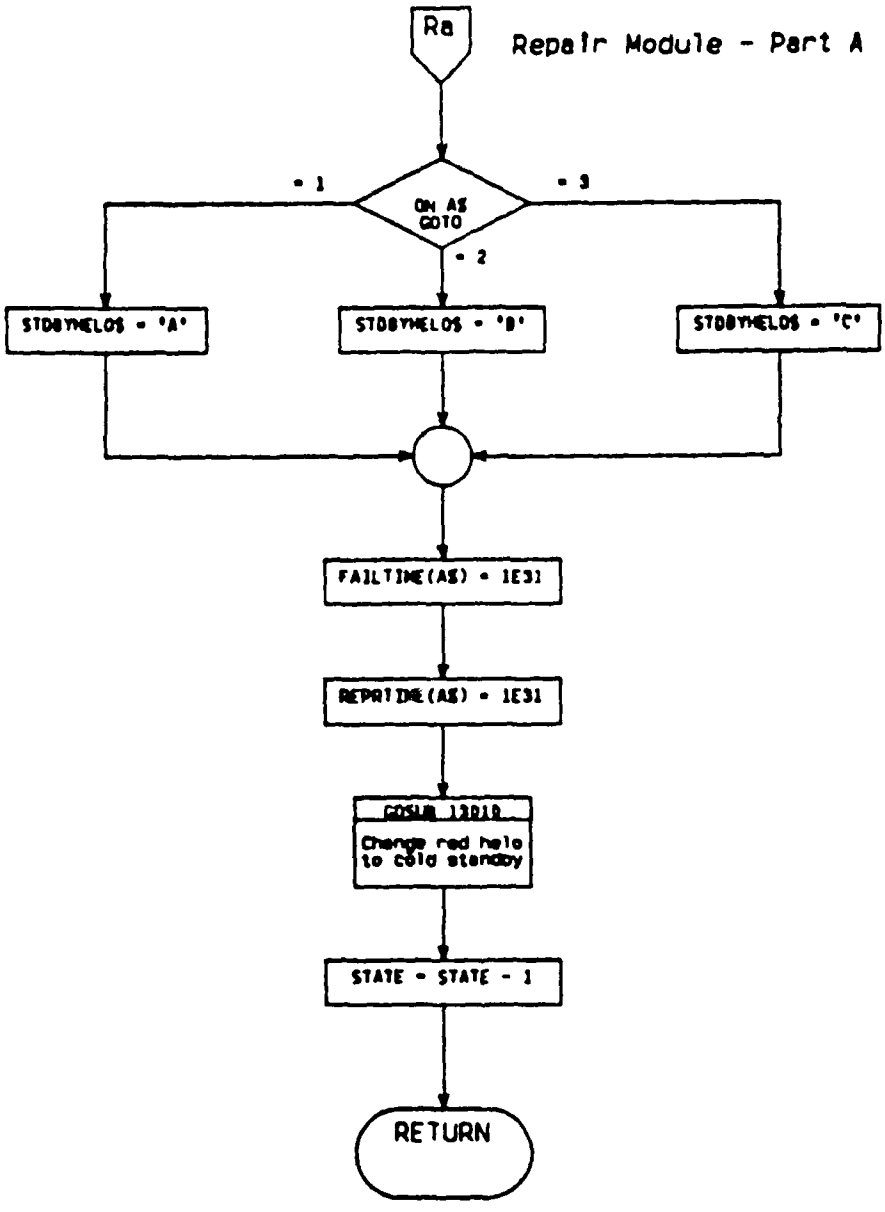


Failure Module Subroutines

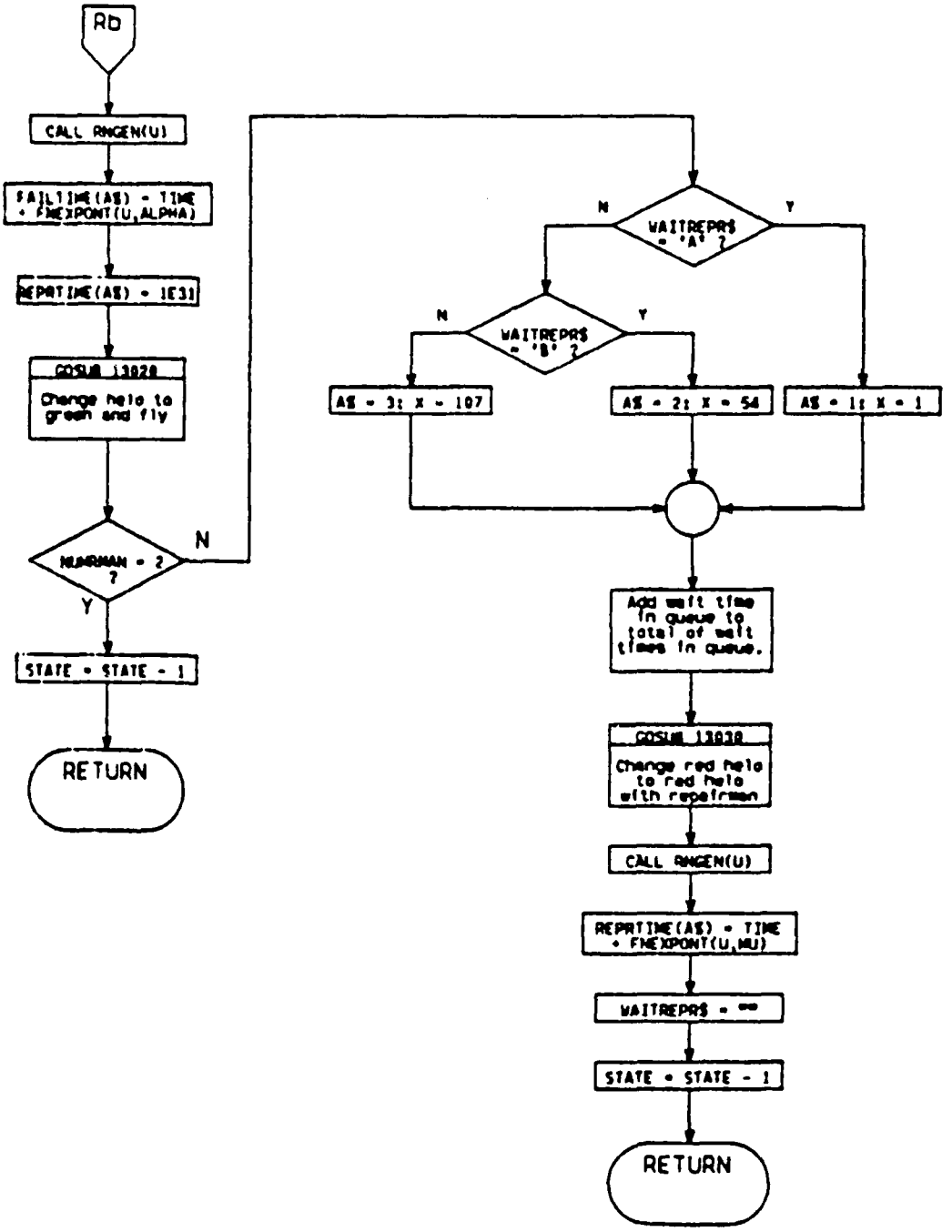


Repair Module

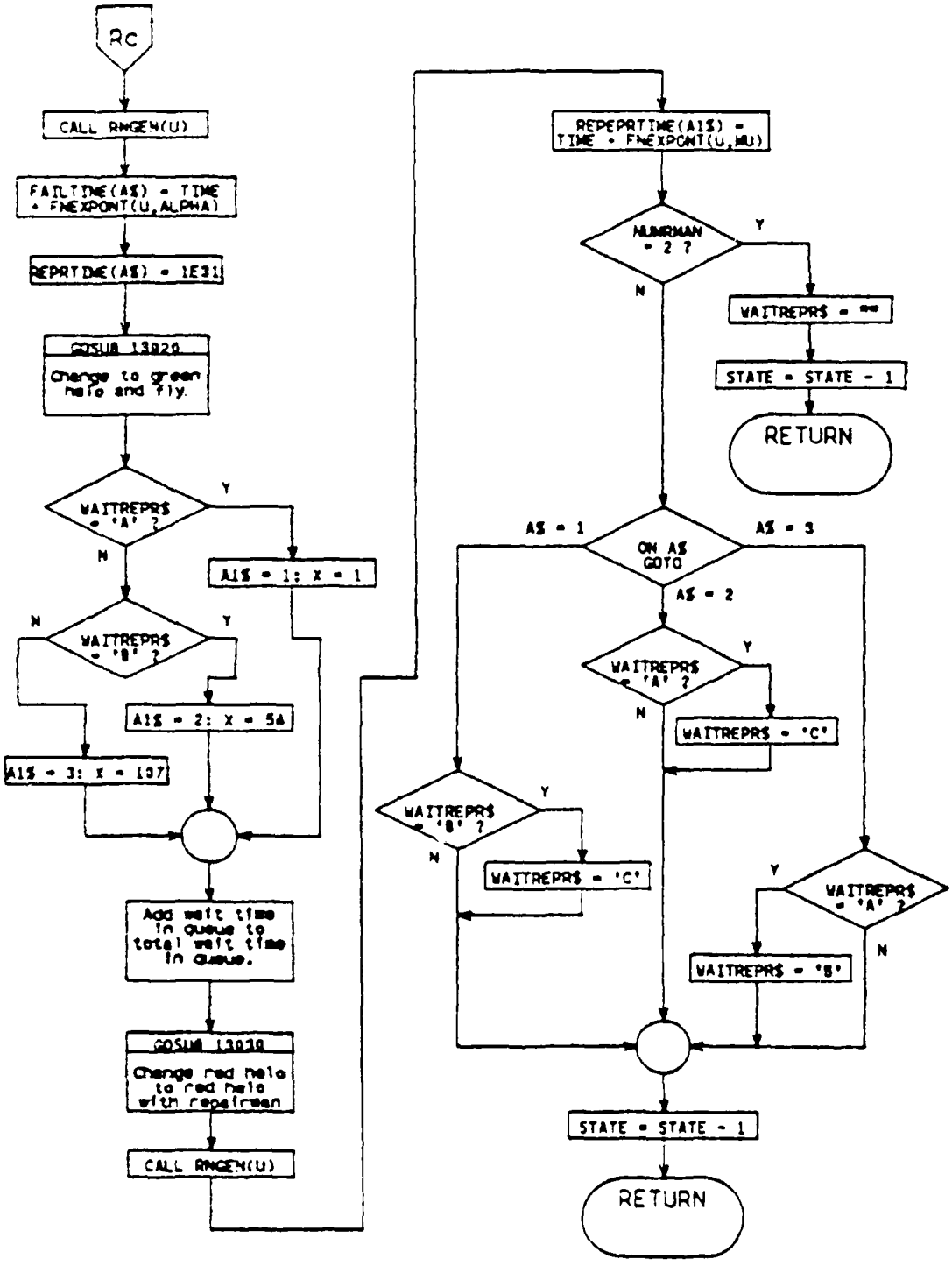



Ra
Repair Module - Part A
= 1
ON AS GOTO
= 3
= 2
STDBYHELO$ = 'A'
STDBYHELO$ = 'B'
STDBYHELO$ = 'C'
FAILTIME(AS) = 1E31
REPRIME(AS) = 1E31
GOSUB 13010
Change red helo to cold standby
STATE = STATE - 1
RETURN

Repair Module - Part B



Repair Module - Part C



```
008C  D1 E0          SAL   AX,1
008E  D1 D2          RCL   DX,1
0090  D1 D3          RCL   BX,1
0092  D1 E0          SAL   AX,1
0094  D1 D2          RCL   DX,1
0096  D1 D3          RCL   BX,1
0098  2B C6          SUB   AX,SI       ;subtract original X
009A  1B D7          SBB   DX,DI
009C  1B D9          SBB   BX,CX
009E  8B F0          MOV   SI,AX       ;saving new X
00A0  8B FA          MOV   DI,DX
00A2  8B CB          MOV   CX,BX
00A4  D1 E0          SAL   AX,1        ;and repeat this 5 times
00A6  D1 D2          RCL   DX,1
00A8  D1 D3          RCL   BX,1
00AA  D1 E0          SAL   AX,1
00AC  D1 D2          RCL   DX,1
00AE  D1 D3          RCL   BX,1
00B0  D1 E0          SAL   AX,1
00B2  D1 D2          RCL   DX,1
00B4  D1 D3          RCL   BX,1
00B6  2B C6          SUB   AX,SI
00B8  1B D7          SBB   DX,DI
00BA  1B D9          SBB   BX,CX
00BC  8B F0          MOV   SI,AX
00BE  8B FA          MOV   DI,DX
00C0  8B CB          MOV   CX,BX
00C2  D1 E0          SAL   AX,1
00C4  D1 D2          RCL   DX,1
00C6  D1 D3          RCL   BX,1
00C8  D1 E0          SAL   AX,1
00CA  D1 D2          RCL   DX,1
00CC  D1 D3          RCL   BX,1
00CE  D1 E0          SAL   AX,1
00D0  D1 D2          RCL   DX,1
00D2  D1 D3          RCL   BX,1
00D4  2B C6          SUB   AX,SI
00D6  1B D7          SBB   DX,DI
00D8  1B D9          SBB   BX,CX
00DA  8B F0          MOV   SI,AX
00DC  8B FA          MOV   DI,DX
00DE  8B CB          MOV   CX,BX
00E0  D1 E0          SAL   AX,1
00E2  D1 D2          RCL   DX,1
00E4  D1 D3          RCL   BX,1
00E6  D1 E0          SAL   AX,1
00E8  D1 D2          RCL   DX,1
00EA  D1 D3          RCL   BX,1
00EC  D1 E0          SAL   AX,1
```

```
002B  2B C6                     SUB     AX,SI
002D  1B D7                     SBB     DX,DI
002F  8B F0                     MOV     SI,AX
0031  8B FA                     MOV     DI,DX
0033  D1 E0                     SAL     AX,1
0035  D1 D2                     RCL     DX,1
0037  D1 E0                     SAL     AX,1
0039  D1 D2                     RCL     DX,1
003B  D1 E0                     SAL     AX,1
003D  D1 D2                     RCL     DX,1
003F  2B C6                     SUB     AX,SI
0041  1B D7                     SBB     DX,DI
0043  8B F0                     MOV     SI,AX
0045  8B FA                     MOV     DI,DX
0047  D1 E0                     SAL     AX,1
0049  D1 D2                     RCL     DX,1
004B  D1 E0                     SAL     AX,1
004D  D1 D2                     RCL     DX,1
004F  D1 E0                     SAL     AX,1
0051  D1 D2                     RCL     DX,1
0053  2B C6                     SUB     AX,SI
0055  1B D7                     SBB     DX,DI
0057  8B F0                     MOV     SI,AX
0059  8B FA                     MOV     DI,DX
005B  D1 E0                     SAL     AX,1
005D  D1 D2                     RCL     DX,1
005F  D1 E0                     SAL     AX,1
0061  D1 D2                     RCL     DX,1
0063  D1 E0                     SAL     AX,1
0065  D1 D2                     RCL     DX,1
0067  2B C6                     SUB     AX,SI
0069  1B D7                     SBB     DX,DI
006B  8B F0                     MOV     SI,AX
006D  8B FA                     MOV     DI,DX
006F  D1 E0                     SAL     AX,1
0071  D1 D2                     RCL     DX,1
0073  D1 E0                     SAL     AX,1
0075  D1 D2                     RCL     DX,1
0077  D1 E0                     SAL     AX,1
0079  D1 D2                     RCL     DX,1
007B  2B C6                     SUB     AX,SI
007D  1B D7                     SBB     DX,DI
007F  E9 012F R                 JMP     UNIF01      ;done since A$X<M
                        ;
0082  33 DB             MULT48: XOR     BX,BX       ;multiply X by 7^5
0084  8B CB                     MOV     CX,BX
0086  D1 E0                     SAL     AX,1        ;by multiplying X by 8
0088  D1 D2                     RCL     DX,1
008A  D1 D3                     RCL     BX,1
```

APPENDIX D

RNGEN.SRT SOURCE-CODE LISTING

```
                        ;*****************************************************
                        ; RNGEN.SRT -- BASIC USR Routine
                        ; by B.O. Shubert
                        ; Modified by R.E. Nelsen
                        ; 2 May 1984
                        ;
                        ; This subroutine generates and returns a uniform
                        ; random-number in BASIC's single-precision format.
                        ; The seed X is not imported nor returned to BASIC,
                        ; but it is automatically updated. The algorithm
                        ; used is:
                        ;          X = AX mod M
                        ;
                        ; with A = 7^5 and M = 2^31 - 1.  RNGEN is called
                        ; from BASIC using the statements:
                        ;            RNGEN = 0
                        ;            DEF SEG = &H1A00
                        ;            CALL RNGEN (U!)
                        ; U! is the returned random number and must be
                        ; defined prior to the first call to RNGEN.
                        ;*****************************************************
                        ;
0000                            CSEG    SEGMENT
                                ASSUME CS:CSEG,DS:CSEG,ES:NOTHING
                        ;
0000                            ORG     0
0000                    RNGEN   PROC    FAR
0000  55                        PUSH    BP              ;save for BASIC
0001  8B EC                     MOV     BP,SP           ;point to args on stack
0003  1E                        PUSH    DS              ;save for BASIC
0004  8C C8                     MOV     AX,CS           ;make data accessable
0006  8E D8                     MOV     DS,AX
                        ;
0008  A1 0161 R                 MOV     AX,LOSEED       ;get old X
000B  8B 16 0163 R              MOV     DX,HISEED
000F  8B F0                     MOV     SI,AX
0011  8B FA                     MOV     DI,DX
0013  83 FA 01                  CMP     DX,1            ;check if A*X < M
0016  77 6A                     JA      MULT48          ;48 bit multiply if not
0018  72 05                     JB      MULT32          ;else 32 bit multiply
001A  3D F31D                   CMP     AX,0F31DH
001D  73 63                     JNB     MULT48
001F  D1 E0                     SAL     AX,1            ;multiply X by 7^5
0021  D1 D2                     RCL     DX,1
0023  D1 E0                     SAL     AX,1
0025  D1 D2                     RCL     DX,1
0027  D1 E0                     SAL     AX,1
0029  D1 D2                     RCL     DX,1
```

```
20147 LOCATE 8,1: PRINT " The numbers listed under the colum    heading 'Limit' are the limiting va
lues as time goes to infinity of the above   defined quantities.  These values are   computed with t
he balance equations for the model."
20148 LOCATE 15,1:PRINT " The values listed under the colum     heading 'Estimate' are the estimate
s of the limiting values.  These quantities  are calculated with data collected from"
20149 PRINT "the simulation. The estimates are recom-puted upon every occurance of an event."
20150 LOCATE 25,2:PRINT "Press any key to return to menu...";:GOSUB 10000:GOTO 120
20151 '
29999 '=====BASICA AND COLOR/GRAPHICS ADAPTER CHECK=========
30000 DEF SEG =0:IF (PEEK(&H410) AND &H30) <> &H30 THEN DEF SEG:GOTO 30007
30003 LOCATE 3,1:PRINT "Sorry..."
30004 PRINT "You do not have the color/graphics monitor adapter!"
30005 PRINT "This simulation uses graphics and requires that adapter."
30006 DEF SEG : END
30007 ON ERROR GOTO 30008: PLAY "p16":GOTO 30011
30008 WIDTH 80:CLS:LOCATE 3,1
30009 PRINT "This simulation uses advanced BASIC."
30010 PRINT "Reload this program after using the command 'BASICA'.":END
30011 ON ERROR GOTO 0: RETURN
30012 '
30019 '=====FILE-LOADING ERROR TRAPPING ROUTINE=============
30020 IF ERR <> 53 THEN ON ERROR GOTO 0
30022 CLS:LOCATE 3,1:PRINT "On which drive can this program be found?":GOSUB 10000
30024 DRVE$=A$+":": RESUME 82
```

```
20044 LOCATE 11,2:PRINT "random-number generator which formed"
20045 LOCATE 13,2:PRINT "the basis for RNGEN.SRT."
20046 LOCATE 25,2:PRINT "Press any key to continue...";:GOSUB 10000:RETURN
20047 '
20099 '=====INSTRUCTIONS==================================
20100 SCREEN 0,1:COLOR 14,1,1:WIDTH 40:CLS
20110 LOCATE 1,4:PRINT "Instructions for the Program Menu";
20111 LOCATE 3,2:PRINT "Press <s> to set the random-number seed";:PRINT "and enter a positive intege
r when asked.The program returns to the menu."
20112 PRINT:PRINT " Press <c> to change model parameters.  The program will ask for the machine's m
ean time to failure (MTTF), the repair-";
20113 PRINT "men's mean time for repair (MTFR), and  the number of repairmen to employ.  The"
20114 PRINT "MTTF and the MTFR should be positive    integers that are greater than 10.  The number
of repairmen can be 1 or 2.  The simulation starts after the number of   repairmen has been entered.
"
20115 LOCATE 18,2: PRINT "Press <d> to use default parameters.";:PRINT "The simulation will immediat
ely start   with the values: MTTF = 15, MTFR = 10,  number of repairmen = 1."
20116 LOCATE 23,2: PRINT "To end the program, press <e>."
20117 LOCATE 25,2: PRINT "Press any key to continue...";:GOSUB 10000
20120 CLS:LOCATE 1,13: PRINT "Keyboard Commands";
20121 LOCATE 4,2: PRINT "During the simulation, you may use the";:PRINT "following keyboard commands
:"
20122 LOCATE 7,5: PRINT "<p> - pause the simulation."
20123 LOCATE 9,5: PRINT "<c> - continue the simulation."
20124 LOCATE 11,5: PRINT "<s> - stop the simulation and return"
20125 LOCATE 13,11: PRINT "to the program menu."
20126 LOCATE 25,2:PRINT "Press any key to continue...";:GOSUB 10000
20130 CLS:LOCATE 1,13: PRINT "Graphic Display";
20131 LOCATE 3,2:PRINT "The variable names in the display are";:PRINT "defined as:";
20132 LOCATE 6,1: PRINT "TIME: current time on simulation clock.";
20133 LOCATE 8,1: PRINT "N(t): no. of machines down at time t.";
20134 LOCATE 10,3: PRINT "AR: average arrival rate of machines";
20135 LOCATE 11,7: PRINT "into the repair queue.";
20136 LOCATE 13,3: PRINT "Wq: average waiting time of machines";
20137 LOCATE 14,7: PRINT "in the repair queue.";
20138 LOCATE 16,3: PRINT "Lq: average length of repair queue.";
20139 LOCATE 18,4: PRINT "W: average down time of a machine.";
20140 LOCATE 20,4: PRINT "L: average no. of down machines.";
20141 LOCATE 22,2: PRINT "Avl: machine availability.";
20142 LOCATE 25,2: PRINT "Press any key to continue...";:GOSUB 10000
20143 CLS:LOCATE 1,3: PRINT "Pd: probability that a down machine";
20144 LOCATE 2,7: PRINT "must wait for repair.";
20145 LOCATE 4,3: PRINT "Ps: probilility that an up machine";
20146 LOCATE 5,7: PRINT "must wait in cold standby.";
```

```
15030 GOSUB 15038
15032 X=131
15034 GOSUB 15038
15036 GOTO 15062
15038 Y1=Y2-4
15040 CIRCLE (X,Y1),12,1,0,PI
15042 DRAW "C1 BU12 U4 L22 R44"
15044 CIRCLE (X,Y2),14,1,7.5*PI/6,11.5*PI/6
15046 DRAW "C1 BD2 L4 NM-2,-15 M-8,-6"
15048 DRAW "D11 M-4,+5 D4 NL1 NR1 U3 M+6,-3"
15050 DRAW "BR20 M+6,+3 D3 NL1 NR1 U4 M-4,-5 U11"
15052 DRAW "M-8,+6 NM+2,-15 L4"
15054 PAINT STEP (0,4),1,1
15056 DRAW "C0 L4 D6 R1 U6 R6 D6 R1 U6 L4"
15058 CIRCLE STEP (0,4),1,0:PSET STEP (0,0),0
15060 RETURN
15062 'Statistics Routine
15064 LOCATE 14,22:PRINT "MTTF:";:LOCATE 14,32:PRINT "MTFR:";
15066 LINE (161,113)-(319,113),2
15068 LOCATE 16,22:PRINT "Var":LOCATE 16,26:PRINT "Estimate":LOCATE 16,35:PRINT "Limit"
15070 LOCATE 18,22:PRINT "AR"
15072 LOCATE 19,22:PRINT "Wq"
15074 LOCATE 20,22:PRINT "Lq"
15076 LOCATE 21,22:PRINT "W"
15078 LOCATE 22,22:PRINT "L"
15080 LOCATE 23,22:PRINT "Avl"
15082 LOCATE 24,4:PRINT "A";:LOCATE 24,10:PRINT "B";
15084 LOCATE 24,17:PRINT "C";:LOCATE 24,22:PRINT "Pd";
15086 LOCATE 25,22:PRINT "Ps";
15087 GET (0,0) - (319,199),MAINSCRN
15088 GET (1,143)-(49,183),GRNHELO
15089 RETURN
15090 '
19999 '=====TITLE SCREEN==================================
20000 SCREEN 0,1: COLOR 14,1,1: WIDTH 40: CLS
20010 LOCATE 2,12: PRINT "GRAPHIC SIMULATION"
20012 LOCATE 4,18: PRINT "of the"
20014 LOCATE 6,9 : PRINT "MACHINE - REPAIRMAN MODEL"
20016 LOCATE 9,14: PRINT "by R.E. Nelsen"
20020 LOCATE 14,5: PRINT "Submitted in partial fulfillment"
20021 LOCATE 15,2: PRINT "of the requirements for the degree of"
20022 LOCATE 17,1: PRINT "Master of Science in Operations Research"
20023 LOCATE 19,4: PRINT "from the Naval Postgraduate School"
20024 LOCATE 20,11: PRINT "Monterey, California"
20025 LOCATE 22,2: PRINT "Advisors: J.D. Esary, A.F. Andrus"
20030 LOCATE 25,2: PRINT "Press any key to continue...";:GOSUB 10000
20040 CLS:LOCATE 1,14: PRINT "Acknowledgment"
20041 LOCATE 5,6:PRINT "The author would like to thank"
20042 LOCATE 7,2:PRINT "Associate Professor Bruno Shubert for"
20043 LOCATE 9,2:PRINT "the donation of his assembly-language"
```

```
12116 PAINT STEP (0,4),2,2
12117 DRAW "C0 L4 D6 R1 U6 R6 D6 R1 U6 L4"
12118 CIRCLE STEP (0,4),2,0:PSET STEP (0,0),0
12120 GET (X,143)- STEP (48,40),REDHELO:X1=X1+19
12130 CIRCLE (X1,165),4,1:PAINT (X1,165),3,1
12131 DRAW "C1 BM-4,-2 NL3 M+7,+1 M-4,-2"
12132 LINE STEP (0,6)- STEP (2,1),3,BF
12133 LINE STEP (1,0)- STEP (-4,6),1,BF
12134 LINE STEP (1,0)- STEP (2,6),1,BF
12135 DRAW "C1 L3 BU9 L2 M-4,-1 U1 M+4,+1 R2":PAINT STEP (-1,1),1,1
12140 GET (X,143)- STEP (48,40),REDHELO2:RETURN
12141 '
12209 =====FAILURE DRAWING SUBROUTINES=====================
12210 PUT (X,105),GRNHELO:PUT (X,143),REDHELO:RETURN:' No repairman
12211 '
12220 PUT (X,105),GRNHELO:PUT (X,143),REDHELO
12221 FOR I=1 TO 300:NEXT:PUT (X,143),REDHELO
12222 PUT (X,143),REDHELO2:RETURN: ' With repairman
12223 '
12230 PUT (X,143),GRNHELO:PUT (X,105),GRNHELO:RETURN:' Cold standby to flying status.
12231 '
13009 '=====REPAIR MODULE SUBROUTINES=====================
13010 PUT (X,143),REDHELO2:PUT (X,143),GRNHELO:RETURN:' To cold standby
13011 '
13020 PUT (X,143),REDHELO2: PUT (X,143),GRNHELO
13021 FOR I=1 TO 200:NEXT: PUT (X,143),GRNHELO
13022 PUT (X,105),GRNHELO: RETURN: 'Repaired and goes flying.
13023 '
13030 FOR I=1 TO 200:NEXT:PUT (X,143),REDHELO: PUT (X,143),REDHELO2: RETURN: 'Assign a repairman.
13031 '
14999 '=====MAIN SCREEN DRAWING SUBROUTINE==================
15000 LINE (0,8)-(319,8),2:LINE (0,10)-(319,10),2
15002 LINE (31,23)-(31,80):FOR Y=27 TO 75 STEP 16:LINE (27,Y)-(31,Y):NEXT
15004 LINE (31,80)-(281,80)
15006 FOR X=36 TO 281 STEP 5
15008 IF (X=81) OR (X=131) OR (X=181) OR (X=231) OR (X=281) THEN LINE (X,80)-(X,85) ELSE LINE (X,80)
-(X,83)
15010 NEXT
15012 LINE (0,100)-(319,100),2:LINE (0,102)-(319,102),2
15014 LINE (161,102)-(161,199),2
15016 LOCATE 1,8:PRINT "MACHINE - REPAIRMAN MODEL"
15018 LOCATE 4,3:PRINT "3":LOCATE 6,3:PRINT "2":LOCATE 8,3:PRINT "1":LOCATE 10,3:PRINT "0"
15020 LOCATE 3,1:PRINT "N(t)":LOCATE 12,32:PRINT "Time:";
15022 'Helocopter drawing routine
15024 PI = 3.141593:X=25:Y2=165
15026 GOSUB 15038
15028 X=78:Y2=127
```

```
3391 '
9998 '***** SUBROUTINES ************************************
9999 '
10000 DEF SEG:POKE &H6A,0: 'General input routine===========
10001 A$= INKEY$:IF A$="" THEN 10001 ELSE RETURN
10002 '
10010 DEF SEG: POKE &H6A,0: 'Pause routine==================
10011 A$= INKEY$: IF A$="c" OR A$="C" THEN RETURN ELSE GOTO 10011
10012 '
10099 '=====STATISTICS PRINT SUBROUTINE=====================
10100 LOCATE 18,Y:PRINT USING FMT1$;LAMBDA!;
10101 LOCATE 19,Y:PRINT USING FMT1$;WQ!;
10102 LOCATE 20,Y:PRINT USING FMT1$;LQ!;
10103 LOCATE 21,Y:PRINT USING FMT1$;W!;
10104 LOCATE 22,Y:PRINT USING FMT1$;L!;
10105 LOCATE 23,Y:PRINT USING FMT1$;AVL!;
10106 LOCATE 24,Y:PRINT USING FMT1$;PDOWN!;
10107 LOCATE 25,Y:PRINT USING FMT1$;PSTDBY!;
10108 RETURN
10109 '
10209 '=====NEXT EVENT SUBROUTINES=========================
10210 IF NXTFAIL!=FAILTIME!(1) THEN FAILINGHELO$="A" ELSE IF NXTFAIL!=FAILTIME!(2) THEN FAILINGHELO$
="B" ELSE FAILINGHELO$="C"
10211 RETURN
10220 IF NXTREPR!=REPRTIME!(1) THEN REPAIREDHELO$="A" ELSE IF NXTREPR!=REPRTIME!(2) THEN REPAIREDHEL
O$="B" ELSE REPAIREDHELO$="C"
10221 RETURN
10222 '
11049 '=====TIC MARK SUBROUTINE============================
11050 AZ=STATE+1:ON AZ GOTO 11051,11052,11053,11054
11051 Y=75:LINE (XX,Y) - STEP (5,0),2:RETURN
11052 Y=59:LINE (XX,Y) - STEP (5,0),2:RETURN
11053 Y=43:LINE (XX,Y) - STEP (5,0),2:RETURN
11054 Y=27:LINE (XX,Y) - STEP (5,0),2:RETURN
11055 '
12099 '=====FIRST FAILURE SUBROUTINE=======================
12100 DEF SEG =&H1A00:CALL RNGEN(U!):A!= FNEXPONT!(U!,MU!)+TIME
12102 IF FAILINGHELO$="B" THEN X=54:X1=78:REPRTIME!(2)=A!:FAILTIME!(2)=1E+31:GOTO 12106
12104 IF FAILINGHELO$="C" THEN X=107:X1=131:REPRTIME!(3)=A!:FAILTIME!(3)=1E+31
12106 PUT (X,105),GRNHELO
12110 PI = 3.141593
12111 CIRCLE (X1,161),12,2,0,PI
12112 DRAW "C2 BU12 U4 M-12,+10 M-8,-4 D10 BM+20,-16 M+12,+10 R6 M+4,+5"
12113 CIRCLE (X1,165),14,2,7.5*PI/6,11.5*PI/6
12114 DRAW "C2 BD2 L4 NM-2,-15 M-8,-6":DRAW "D11 M-4,+5 D4 NL1 NR1 U3 M+6,-3"
12115 DRAW "BR20 M+6,+3 D3 NL1 NR1 U4 M-4,-5 U11":DRAW "M-8,+6 NM+2,-15 L4"
```

```
2300 REPRTIME!(AZ) = 1E+31:FAILTIME!(AZ) = 1E+31
2310 IF NUMRMAN=2 THEN WAITREPR$= MID$("ABC",AZ,1)
2315 STARTQUE(AZ) = TIME
2320 GOSUB 12210: 'Ground flying helo: no repairman.
2330 RETURN
2331 '
2999 '=====REPAIR SUBROUTINE================================
3000 AZ= INSTR("ABC",REPAIREDHELO$)
3010 X = FNHELOPOSIT(REPAIREDHELO$)
3020 ON STATE GOTO 3100,3200,3300
3021 '
3100 STDBYHELO$= MID$("ABC",AZ,1)
3110 FAILTIME!(AZ) = 1E+31
3120 REPRTIME!(AZ) = 1E+31
3130 GOSUB 13010: 'Change red helo to cold stand-by helo.
3140 STATE = STATE-1: RETURN
3141 '
3200 DEF SEG =&H1A00:CALL RNGEN(U!)
3210 FAILTIME!(AZ) = FNEXPONT!(U!,ALPHA!)+TIME
3212 REPRTIME!(AZ) = 1E+31
3220 GOSUB 13020: 'Change red helo to green and fly.
3230 IF NUMRMAN=2 THEN STATE=STATE-1: RETURN
3240 AZ= INSTR("ABC",WAITREPR$)
3242 X = FNHELOPOSIT(WAITREPR$)
3244 SUMWQ = SUMWQ+TIME-STARTQUE(AZ)
3250 GOSUB 13030: 'Change red helo to red helo w/ repairman.
3260 CALL RNGEN(U!):REPRTIME!(AZ)= FNEXPONT!(U!,MU!)+TIME
3262 WAITREPR$=""
3270 STATE=STATE-1: RETURN
3271 '
3300 DEF SEG =&H1A00:CALL RNGEN(U!)
3310 FAILTIME!(AZ)= FNEXPONT!(U!,ALPHA!)+TIME
3312 REPRTIME!(AZ)= 1E+31
3320 GOSUB 13020: 'Change to green helo and fly.
3330 A1Z= INSTR("ABC",WAITREPR$)
3332 X= FNHELOPOSIT(WAITREPR$): 'Helo at head of queue.
3334 SUMWQ = SUMWQ+TIME-STARTQUE(A1Z)
3340 GOSUB 13030: 'Change red helo to red helo w/ repairman.
3342 CALL RNGEN(U!)
3344 REPRTIME!(A1Z)= FNEXPONT!(U!,MU!)+TIME
3346 IF NUMRMAN=2 THEN WAITREPR$="":GOTO 3390
3350 ON AZ GOTO 3362,3372,3382 :' Advance the queue.
3360 'REPAIREDHELO$ = "A"
3362 IF WAITREPR$="B" THEN WAITREPR$="C" ELSE WAITREPR$="B"
3364 GOTO 3390
3370 'REPAIREDHELO$ = "B"
3372 IF WAITREPR$="A" THEN WAITREPR$="C" ELSE WAITREPR$="A"
3374 GOTO 3390
3380 'REPAIREDHELO$ = "C"
3382 IF WAITREPR$="A" THEN WAITREPR$="B" ELSE WAITREPR$="A"
3390 STATE=STATE-1: RETURN
```

```
1302 LASTEVNT = NXTEVNT
1310 IF NXTFAIL!<=NXTREPR! THEN GOSUB 2000 ELSE GOSUB 3000
1311 '
1399 '** Update statistical estimates **
1400 LAMBDA!=SUMFAIL/TIME
1410 IF NUMRMAN=1 THEN PDOWN!=(SUMP!(2)+SUMP!(3))/TIME:LQ!=PDOWN!+SUMP!(3)/TIME ELSE LQ!=SUMP!(3)/TI
ME:PDOWN!=LQ!
1420 L!=(SUMP!(1)+SUMP!(2)+SUMP!(2)+SUMP!(3)+SUMP!(3)+SUMP!(3))/TIME
1430 WQ!=SUMWQ/SUMFAIL:W!=WQ!+1/MU!:PSTDBY!=SUMP!(0)/TIME
1440 AVL!=SUMP!(0)/TIME+(SUMP!(1)+SUMP!(1)+SUMP!(2))/(TIME+TIME+TIME)
1450 Y=25:GOSUB 10100: 'Print estimates to screen
1451 '
1490 A$= INKEY$:IF A$="" THEN 1500
1491 IF A$="s" OR A$="S" THEN 120
1493 IF A$="p" OR A$="P" THEN GOSUB 10010
1495 DEF SEG: POKE &H6A,0: 'Clear key buffer
1500 WEND :'No stop command so go back to line 1100.
1510 CLS: LOCATE 12,1: PRINT "Program time limit reached."
1512 PRINT "Press any key to return to menu..."
1514 GOSUB 10000: GOTO 120
1515 '
1999 '=====FAILURE MODULE==================================
2000 SUMFAIL=SUMFAIL+1:STATE = STATE + 1
2020 IF FIRSTFAILURE THEN GOSUB 12100:FIRSTFAILURE=FALSE:GOTO 2130
2030 A%= INSTR("ABC",FAILINGHELO$)
2032 X= FNHELOPOSIT(FAILINGHELO$)
2040 ON STATE GOTO 2100,2200,2300
2041 '
2100 DEF SEG =&H1A00:CALL RNGEN(U!)
2110 REPRTIME!(A%)= FNEXPONT!(U!,MU!)+TIME
2112 FAILTIME!(A%)= 1E+31
2120 GOSUB 12220: 'Ground flying helo & assign repairman.
2130 A%= INSTR("ABC",STDBYHELO$)
2132 X= FNHELOPOSIT(STDBYHELO$)
2140 CALL RNGEN(U!)
2142 FAILTIME!(A%)= FNEXPONT!(U!,ALPHA!)+TIME
2150 GOSUB 12230: ' Move stand-by helo to flying status.
2160 STDBYHELO$="":RETURN
2161 '
2200 IF NUMRMAN =2 THEN 2240
2210 GOSUB 12210: 'Ground flying helo: no repairman.
2220 WAITREPR$ = MID$("ABC",A%,1)
2222 REPRTIME!(A%) = 1E+31:FAILTIME!(A%) = 1E+31
2224 STARTQUE(A%) = TIME
2230 RETURN
2240 GOSUB 12220: 'Ground flying helo & assign a repairman.
2250 DEF SEG =&H1A00: CALL RNGEN(U!)
2252 REPRTIME!(A%) = FNEXPONT!(U!,MU!)+TIME
2254 FAILTIME!(A%) = 1E+31: WAITREPR$=""
2260 RETURN
2261 '
```

```
520 LOCATE 12,37:PRINT USING FMT2$;TIME;
521 GOTO 1000
522 '
899 '=====PROGRAM END ROUTINE==============================
900 SCREEN 0,0:WIDTH 80:COLOR 6:PRINT "Program ended...":END
998 '
999 '=====CLOCK MODULE=======================================
1000 FAILTIME!(1)=1E+31: 'Calculate 1st failure time (B & C)
1010 DEF SEG =&H1A00:CALL RNGEN(U!)
1012 FAILTIME!(2)= FNEXPONT!(U!,ALPHA!)+TIME
1020 CALL RNGEN(U!)
1022 FAILTIME!(3)= FNEXPONT!(U!,ALPHA!)+TIME
1030 REPRTIME!(1)=1E+31: '1E+31 = "infinity"
1032 REPRTIME!(2)=1E+31
1034 REPRTIME!(3)=1E+31
1036 STDBYHELO$="A"
1099 '** Determine next event **
1100 WHILE NXTEVNT<9950: 'Sets upper limit to prgm run time.
1110 IF NOT STATE=0 THEN 1120
1112 NXTFAIL!= FNMIN!(FAILTIME!(1),FAILTIME!(2),FAILTIME!(3))
1114 NXTREPR! = 1E+31
1116 GOSUB 10210:GOTO 1180
1120 IF NOT STATE=3 THEN 1130
1122 NXTFAIL!= 1E+31
1124 NXTREPR!= FNMIN!(REPRTIME!(1),REPRTIME!(2),REPRTIME!(3))
1126 GOSUB 10220:GOTO 1180
1130 NXTFAIL!= FNMIN!(FAILTIME!(1),FAILTIME!(2),FAILTIME!(3))
1132 NXTREPR!= FNMIN!(REPRTIME!(1),REPRTIME!(2),REPRTIME!(3))
1134 GOSUB 10210
1136 GOSUB 10220
1180 IF NXTFAIL!<=NXTREPR! THEN 1184
1182 NXTEVNT = FNROUNDUP(NXTREPR!):GOTO 1200
1184 NXTEVNT = FNROUNDUP(NXTFAIL!)
1185 '
1199 '** Run clock and update time chart **
1200 DEF SEG:POKE &H6A,0: 'Clear key buffer
1210 WHILE TIME < NXTEVNT
1220 TIME=TIME+1
1230 IF TIME<=50 THEN GOSUB 11050:XX=XX+5:FOR I=1 TO 900:NEXT:GOTO 1240
1232 DEF SEG = &H1A00
1234 CALL SCRNSHIFT
1238 XX=276:GOSUB 11050
1240 LOCATE 12,37:PRINT USING FMT2$;TIME;
1241 A$=INKEY$ :IF A$="" THEN 1260
1242 IF A$="s" OR A$="S" THEN 120
1244 IF A$="p" OR A$="P" THEN GOSUB 10010
1250 DEF SEG:POKE &H6A,0: 'Clear key buffer
1260 WEND
1261 '
1299 '** Update display and determine next event **
1300 SOUND 120,3:SUMP!(STATE)=SUMP!(STATE)+TIME-LASTEVNT
```

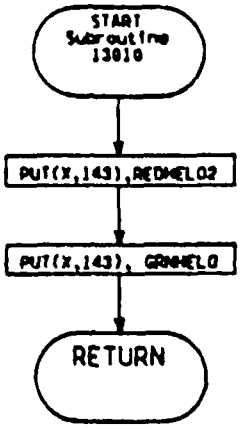
```
159 DEF SEG = &H1A00
160 A1!= INT (A#/16777210#):A2!= INT((A#-A1!*16777210#)/65536!)
162 'Poke the seed's upper 2 bytes into RNGEN's seed storage.
163 POKE &H164,A1!: POKE &H163,A2!
164 A#=A#-A1!*16777210#-A2!*65536!:A1!= INT(A#/256):A2!= A#-A1!*256
165 'Poke the seed's lower 2 bytes into RNGEN's seed storage.
166 POKE &H162,A1!: POKE &H161,A2!
167 GOTO 120: 'Return to menu
168 '
199 '=====PARAMETERS ROUTINE================================
200 COLOR 15,5,0:CLS:MSG$=">>Enter a positive integer only.<<"
202 LOCATE 1,6:PRINT "* CHANGE MODEL PARAMETERS *":PRINT SG$
203 LOCATE 4,1:INPUT "Enter Mean Time To Failure...",A!:IF A!<=0 THEN LOCATE 23,3:PRINT MSG$;:LOCATE
 4,1:PRINT STRING$(40,32);:GOTO 203 ELSE ALPHA!=1/A!
204 LOCATE 6,1:INPUT "Enter Mean Time For Repair...",A!:IF A!<=0 THEN LOCATE 23,3:PRINT MSG$;:LOCATE
 6,1:PRINT STRING$(40,32);:GOTO 204 ELSE MU!=1/A!
205 LOCATE 23,1:PRINT STRING$(40,32);:MSG$=""
206 LOCATE 8,1:INPUT "Enter number of repairmen (1 or 2)...",NUMRMAN:IF (NUMRMAN<>1) AND (NUMRMAN<>2
) THEN LOCATE 8,1:PRINT STRING$ (40,32);:GOTO 206 ELSE 250
210 ALPHA!=1/15:MU!=1/10:NUMRMAN=1
250 NXTFAIL!=0:NXTREPR!=0:FIRSTFAILURE=TRUE:NXTEVNT=0:TIME=0:SUMFAIL=0:LASTEVNT=0:STATE=0:XX=31:SUMW
Q=0
252 FOR I=0 TO 3:SUMP!(I)=0:STARTQUE(I)=0:NEXT
260 '
299 '=====THEORETICAL STATISTICS ROUTINE====================
300 P!(1)=2*ALPHA!/MU!
320 IF NUMRMAN=1 THEN P!(2)=P!(1)*P!(1):P!(3)=P!(2)*ALPHA!/MU! ELSE P!(2)=P!(1)*ALPHA!/MU!:P!(3)=P!(
2)*ALPHA!/(MU!+MU!)
324 P!(0)=1/(1+P!(1)+P!(2)+P!(3))
325 P!(1)=P!(1)*P!(0)
326 P!(2)=P!(2)*P!(0)
327 P!(3)=P!(3)*P!(0)
330 IF NUMRMAN=1 THEN PDOWN!=P!(2)+P!(3):LQ!=PDOWN!+P!(3) ELSE LQ!=P!(3):PDOWN!=P!(3)
340 L!=P!(1)+2*P!(2)+3*P!(3)
350 LAMBDA!=(2*P!(0)+2*P!(1)+P!(2))*ALPHA!
360 WQ!=LQ!/LAMBDA!
370 W!=L!/LAMBDA!
380 PSTDBY!=P!(0)
390 AVL!=P!(0)+(P!(1)+P!(1)+P!(2))/3
395 '
499 '=====PRINT MAIN DISPLAY ROUTINE=======================
500 SCREEN 1,0:COLOR 9,0
502 IF FIRSTSCRN THEN GOSUB 15000:FIRSTSCRN=FALSE:GOTO 510
505 PUT (0,0),MAINSCRN
510 LOCATE 14,27:PRINT USING FMT2$;1/ALPHA!;
511 LOCATE 14,37:PRINT USING FMT2$;1/MU!;
512 Y=33:GOSUB 10100: ' Print statistics to screen.
```

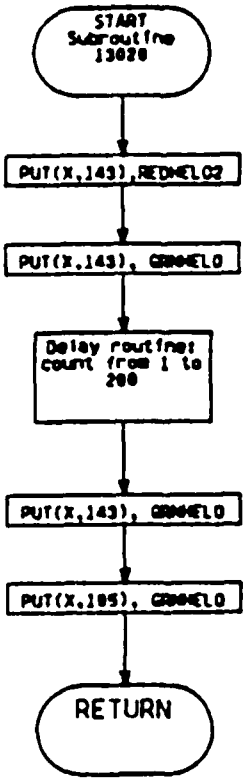
APPENDIX C

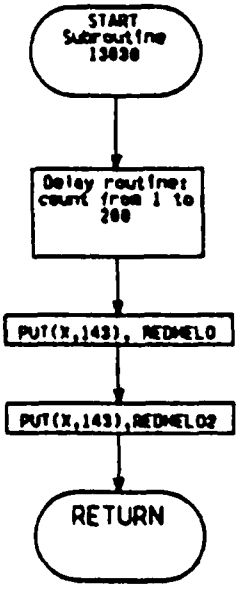
MACHREPR.BAS PROGRAM LISTING

```
0 'MACHREPR.BAS, Ver 3.1, R. E. Nelsen, 30 Aug 1984
1 CLEAR : KEY OFF: CLS
2 GOSUB 30000: 'Check for BASICA and color/graphics adaptor
3 DEFINT A-Z
4 DIM P!(3),SUMP!(3),GRNHELO(268),REDHELO(268),REDHELO2(268)
5 DIM TIMECHRT(1609),MAINSCRN(8002)
6 DIM FAILTIME!(3),REPRTIME!(3),STARTQUE(4)
7 RNGEN=0:TRUE=-1:FALSE=0:U!=0:SCRNSHIFT=384
8 FMT1$="###.###":FMT2$="####":DRVE$="A:"
9 FIRSTSCRN=TRUE:WAITREPR$="":FAILINGHELO$="":STDBYHELO$=""
10 REPAIREDHELO$="":SG$ = STRING$(40,223)
50 DEF FNEXPONT!(A1!,A2!) = - LOG(A1!)/A2!
51 DEF FNROUNDUP(A!) = INT(A!) + 1
52 DEF FNMIN!(A!,B!,C!)=(-(A!<B! AND A!<C!)*A!)+(-(B!<A! AND B!<C!)*B!)+(-(C!<A! AND C!<B!)*C!)
53 DEF FNHELOPOSIT(A$)=-( INSTR("ABC",A$)=1)+(-( INSTR("ABC",A$)=2)*54)+(-( INSTR("ABC",A$)=3)*107)
55 '
60 GOSUB 20000: 'Title screen
80 ON ERROR GOTO 30020: 'Load random number generator.
82 DEF SEG = &H1A00: BLOAD DRVE$+"RNGEN.SRT",0
84 ON ERROR GOTO 0
85 '
90 BLOAD DRVE$+"SCRNSHFT.SRT",384: 'Load screen shift subroutine
91 '
100 '=====MAIN PROGRAM MENU==================================
120 SCREEN 0,1:COLOR 14,3,0:WIDTH 40:CLS
121 LOCATE 7,14:PRINT "PROGRAM MENU":PRINT SG$
122 LOCATE 9,1:PRINT "(I)nstructions."
123 LOCATE 11,1:PRINT "(C)hange model parameters."
124 PRINT:PRINT "(D)efault model parameters."
125 PRINT:PRINT "(S)et the random number generator seed."
126 PRINT:PRINT "(E)nd the program.": PRINT SG$
130 LOCATE 21,2: PRINT "Enter your selection...";
131 '
140 GOSUB 10000:A%= INSTR("ICDSEicdse",A$)
141 IF A%=0 THEN 120 ELSE ON A% GOTO 20100,200,210,150,900,20100,200,210,150,900
142 '
149 '=====RANDOM NUMBER GENERATOR SEED ROUTINE==============
150 COLOR 14,0,0:CLS
151 LOCATE 1,3: PRINT "* SET RANDOM NUMBER GENERATOR SEED *": PRINT SG$
152 LOCATE 4,2:PRINT "Permissible seed values are integers"
153 PRINT " in the range: 1 to 2147483646."
154 LOCATE 7,2:INPUT "Enter the seed value...";A#
155 IF A#<1 OR A#>2147483646# THEN LOCATE 7,2: PRINT STRING$(39,32): GOTO 154
```

Repair Module Subroutines







```
00EE  D1 D2                     RCL     DX,1
00F0  D1 D3                     RCL     BX,1
00F2  2B C6                     SUB     AX,SI
00F4  1B D7                     SBB     DX,DI
00F6  1B D9                     SBB     BX,CX
00F8  8B F0                     MOV     SI,AX
00FA  8B FA                     MOV     DI,DX
00FC  8B CB                     MOV     CX,BX
00FE  D1 E0                     SAL     AX,1
0100  D1 D2                     RCL     DX,1
0102  D1 D3                     RCL     BX,1
0104  D1 E0                     SAL     AX,1
0106  D1 D2                     RCL     DX,1
0108  D1 D3                     RCL     BX,1
010A  D1 E0                     SAL     AX,1
010C  D1 D2                     RCL     DX,1
010E  D1 D3                     RCL     BX,1
0110  2B C6                     SUB     AX,SI
0112  1B D7                     SBB     DX,DI
0114  1B D9                     SBB     BX,CX       ;BX,DX,AX, now contain A*X
0116  D1 E2                     SAL     DX,1
0118  D1 D3                     RCL     BX,1        ;get K=A*X/2^31 to BX
011A  D1 EA                     SHR     DX,1        ;and X'=A*X(mod 2^31) to DX,AX
011C  03 C3                     ADD     AX,BX       ;calculate X'' = X'+K
011E  83 D2 00                  ADC     DX,0
0121  81 FA 7FFF                CMP     DX,7FFFH    ;see if X'' < M
0125  77 03                     JA      SUBTRM
0127  EB 06 90                  JMP     UNIF01      ;done if so since X'' can't = M
                        ;
012A  40                SUBTRM: INC     AX          ;else subtract M by adding 1
012B  81 DA 8000                SBB     DX,8000H    ;and subtracting 2^31
                        ;
012F  A3 0161 R         UNIF01: MOV     LOSEED,AX   ;store new X
0132  89 16 0163 R              MOV     HISEED,DX
0136  D1 E0                     SAL     AX,1        ;Divide X by 2^31 to make
0138  D1 D2                     RCL     DX,1        ;a uniform (0,1).
013A  32 C9                     XOR     CL,CL       ;Set a counter to zero.
013C  D1 E0             FNDEXP: SAL     AX,1        ;Shift left and count the
013E  D1 D2                     RCL     DX,1        ;number of leading zeros.
0140  72 04                     JC      EXPONT      ;Leading 1 found.
0142  FE C1                     INC     CL          ;Not found, add 1 to counter
0144  EB F6                     JMP     FNDEXP      ;and look at next bit.
0146  D1 EA             EXPONT: SHR     DX,1        ;Shift back to correct value
0148  D1 D8                     RCR     AX,1        ;and suppress leading 1.
014A  B3 80                     MOV     BL,80H      ;Compute normalized exponent.
014C  2A D9                     SUB     BL,CL
                        ;
                        ; BL, DX, and AX now contain a single precision, uniform
                        ; (0,1) random number in BASIC's floating point format.
```

```
                        ;
014E  1F                        POP     DS              ;restore segment register
014F  8B 7E 06                  MOV     DI,[BP]+6       ;get addr of variable U
0152  88 25                     MOV     [DI],AH         ;pass LSB of mantissa
0154  88 55 01                  MOV     [DI]+1,DL       ;pass MSB of mantissa
0157  88 75 02                  MOV     [DI]+2,DH       ;pass HSB of mantissa
015A  88 5D 03                  MOV     [DI]+3,BL       ;pass exponent
015D  5D                        POP     BP              ;restore BP for BASIC
015E  CA 0002                   RET     2               ;FAR return to BASIC
                        ;
0161  4130              LOSEED  DW      4130H           ;storage for X
0163  00AB              HISEED  DW      0ABH
0165                    RNGEN   ENDP
0165                            CSEG    ENDS
                                END
```

APPENDIX E

SCRNSHFT.SRT SOURCE-CODE LISTING

```
                        ;*********************************************
                        ;SCRNSHFT.SRT, Ver 2 -- BASIC USR Routine
                        ;by R.E. Nelsen -- 27 July 1984
                        ;
                        ;This subroutine shifts 5 columns to the left
                        ;the 4 rows of pixels corresponding to the
                        ;model s state vs time graph. It is called
                        ;from BASIC using the commands:
                        ;
                        ;     SCRNSHFT = 0
                        ;     DEF SEG = &H1A00
                        ;     CALL SCRNSHFT
                        ;
                        ;This subroutines calls the ROM BIOS Type 10H
                        ;interrupt (Video I/O) using routines 12 & 13.
                        ;*********************************************
                        ;
0000                            CSEG  SEGMENT
                                ASSUME CS:CSEG, DS:NOTHING
                        ;
                        ;Establish a file header for use by the
                        ;BASIC BLOAD command.
                        ;
0000                    HEADER:
0000  FD                         DB    0FDH      ;Code for BLOAD file
0001  0000                       DW    0         ;Seg addr location
0003  0000                       DW    0         ;Offset location
0005  0086                       DW    RTN_LEN   ;Routine length
                        ;
0007                    SCRN_SHFT PROC  FAR
0007  55                         PUSH  BP        ;Save for BASIC
0008  FB                         STI             ;Enable interrupts
0009  B9 0024                    MOV   CX,36     ;Scrn column 36
000C  BA 001B           AGAIN:   MOV   DX,27     ;Scrn row 27
000F  B4 0D                      MOV   AH,13     ;Read pix at row DX
0011  CD 10                      INT   10H       ;& col CX. Pix in AL
0013  83 E9 05                   SUB   CX,5      ;Shift left 5 col's
0016  B4 0C                      MOV   AH,12     ;Set = 12 to write
0018  CD 10                      INT   10H       ;pixel to new pos'n
001A  83 C1 05                   ADD   CX,5      ;Shift right 5 col's
001D  83 C2 10                   ADD   DX,16     ;Row 43
0020  B4 0D                      MOV   AH,13     ;Set = 13 to read
0022  CD 10                      INT   10H       ;Read pixel
0024  83 E9 05                   SUB   CX,5      ;Shift left 5
0027  B4 0C                      MOV   AH,12     ;Set = 12
0029  CD 10                      INT   10H       ;Write pixel
002B  83 C1 05                   ADD   CX,5      ;Shift right 5 col's
```

```
                        ;
                        ;Now repeat procedure for remaining two rows.
                        ;
002E  83 C2 10                  ADD   DX,16     ;Row 59
0031  B4 0D                     MOV   AH,13
0033  CD 10                     INT   10H
0035  83 E9 05                  SUB   CX,5
0038  B4 0C                     MOV   AH,12
003A  CD 10                     INT   10H
003C  83 C1 05                  ADD   CX,5
003F  83 C2 10                  ADD   DX,16     ;Row 75
0042  B4 0D                     MOV   AH,13
0044  CD 10                     INT   10H
0046  83 E9 05                  SUB   CX,5
0049  B4 0C                     MOV   AH,12
004B  CD 10                     INT   10H
004D  83 C1 06                  ADD   CX,6      ;Shift to new column
0050  81 F9 011A                CMP   CX,282    ;Is last col reached?
0054  72 B6                     JB    AGAIN     ;If no, do next col.
                        ;
                        ;Finished with rows, so now must blank out last
                        ;5 columns of each row.
                        ;
0056  41                        INC   CX        ;Col 283
0057  B4 0D                     MOV   AH,13     ;Set to read dot
0059  CD 10                     INT   10H       ;Read blank dot
005B  8A D8                     MOV   BL,AL     ;Save color code
005D  B9 0115                   MOV   CX,277    ;Col 277
0060  BA 001B           NEW_COL: MOV   DX,27     ;Row 27
0063  B4 0C                     MOV   AH,12     ;Set to write dot
0065  8A C3                     MOV   AL,BL     ;Set backgrnd color
0067  CD 10                     INT   10H       ;Write blank dot
0069  83 C2 10                  ADD   DX,16     ;Do again for
006C  B4 0C                     MOV   AH,12     ;row 43,
006E  8A C3                     MOV   AL,BL
0070  CD 10                     INT   10H
0072  83 C2 10                  ADD   DX,16     ;and row 59,
0075  B4 0C                     MOV   AH,12
0077  8A C3                     MOV   AL,BL
0079  CD 10                     INT   10H
007B  83 C2 10                  ADD   DX,16     ;and row 75.
007E  B4 0C                     MOV   AH,12
0080  8A C3                     MOV   AL,BL
0082  CD 10                     INT   10H
0084  41                        INC   CX        ;New col on right
0085  81 F9 011A                CMP   CX,282    ;Last column?
0089  72 D5                     JB    NEW_COL   ;No, so do next col.
                        ;
                        ;Done, so recover BP and return to BASIC.
```

```
                        ;
008B  5D                        POP   BP
008C  CB                        RET
008D                    SCRN_SHFT  ENDP
= 0086                  RTN_LEN    EQU   $ - SCRN_SHFT
008D  1A                        DB    01AH      ;Need for BLOAD
008E                        CSEG  ENDS
                                END   HEADER
```

BIBLIOGRAPHY

Cheney, W. and Kincaid, D., Numerical Mathematics and Computing, Brook/Cole, 1980.

Coffron, J.W., Programming the 8086/8088, Sybex, 1983.

Davison, R.J., Graphic Simulation of the Poisson Process, Master's Thesis, Naval Postgraduate School, October 1982.

Howson, H.R., "POKEing Around in the IBM PC, Part 1: Accessing System and Hardware Facilities," BYTE, v. 8, November 1983.

Howson, H.R., "POKEing Around in the IBM PC, Part 2: Developing Subroutines for BIOS Interface and Screen-Display Disk Storage," BYTE, v. 8, December 1983.

IBM Corporation Personal Computer Library, BASIC, May 1982.

IBM Corporation Personal Computer Library, Macro-Assembler, May 1982.

IBM Corporation Personal Computer Library, PCDOS 2.10, September 1983.

IBM Corporation Personal Computer Library, Technical Reference, July 1982.

Law, A.M. and Kelton, W.D., Simulation Modeling and Analysis, McGraw-Hill, 1982.

Naval Postgraduate School Report NPS55LW73061A, Random Number Package LLRANDOM, by G.P. Learmonth and P.A.W. Lewis June 1973.

Rollins, D., "The 8088 Connection: Interfacing IBM PC BASIC to Machine-Language Programs," BYTE, v. 8, July 1983.

Ross, S.M., Introduction to Probability Models, 2d ed., Academic Press, 1980.

Scanlon, L.J., IBM PC Assembly Language: A Guide for Programmers, Brady, 1983.

Stein, M.L. and Munro, W.D., Introduction to Machine Arithmetic, Addison-Wesley, 1971.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Colonel F.R. Giordano, U.S. Army United States Military Academy, Code MADN-A West Point, New York 10996	1
4.	Major R.E. Nelsen, U.S. Marine Corps Headquarters Battalion Headquarters, U.S. Marine Corps Henderson Hall Arlington, Virginia 22214	2
5.	Doctor T.J. Schonfeld Traffic Theory Department Bell Laboratories, Room 2M-626 Holmdel, New Jersey 07733	1
6.	Prof. J. D. Esary Code 55Ey Dept. of Operations Research Naval Postgraduate School Monterey, California 93943	2

END

FILMED

5-85

DTIC